intel

APPLICATION NOTE

AP-132

June 1982

Designing Memory Systems with the 8K x 8 iRAM

John J. Fallin
William H. Righter
Memory Components
Applications Engineering

Order Number: 210443-001

1 INTRODUCTION

1.1 RAM Overview

Matching the correct RAM to microprocessors is fundamental to effective product design. Understanding the advantages and disadvantages of each device type enables a microprocessor system designer to choose the best product for his particular design objective.

Two basic types of semiconductor random access memories (RAMs) are in use at present: static RAMs (SRAMs) and dynamic RAMs (DRAMs). Where large amounts of memory at the lowest cost per bit is required, such as main computer memory, the dynamic RAM holds a commanding position. The extra costs of refresh, timing and arbitration overhead are spread over a very large amount of memory. The static RAM, however, provides a better solution for relatively small memory systems where high performance or simple system design is desired.

A major advantage of dynamic RAMs is low memory component cost. A DRAM uses a simple one-transistor, one-capacitor cell for binary storage. This simple design achieves high integration density and low cost. When a DRAM cell is not being written, read or refreshed, it consumes almost no current. At any given time, the majority of the cells in a DRAM array will be in this condition — yielding low overall power consumption.

One disadvantage of DRAMs are their extensive control and interface requirements. The DRAM control circuitry must generate signals such as $\overline{RAS}$ and $\overline{CAS}$, provide refresh cycles, and handle arbitration. This adds to the component count and overhead costs, both in design and implementation.

Conversely, static RAMs need very little external control circuitry and they interface easily to most microprocessors. An SRAM has no refresh requirement and usually has all of its control signals generated directly by the system microprocessor. A disadvantage of the SRAM is its high cell complexity. A typical static RAM cell requires four to six transistors — resulting in a lower cell density and higher manufacturing cost/bit than DRAMs.

A new type of RAM has now been developed that combines the best features of the SRAM and DRAM and is called the iRAM (integrated RAM). An iRAM is an entire dynamic RAM system integrated onto a single silicon chip, including the memory array, refresh logic, arbitration, and control logic. This new implementation combines the cost, power and density advantages of a DRAM with the ease of use of a static RAM. Because all of the DRAM control logic is internal, the memory system can operate autonomously, controlling its own refresh and arbitration. This greatly simplifies microprocessor interfacing and minimizes additional TTL hardware support. Proper refresh is guaranteed and overall system performance improved.

1.2 iRAM Concept Background

With the advent of VLSI technology and 64K RAM densities, it became possible to further integrate and simplify memory system design. LSI memory controllers integrate all of these components into a single device (such as Intel's 8202A and 8203 dynamic DRAM controllers). Figure 1 shows the major elements of such a dynamic RAM controller.

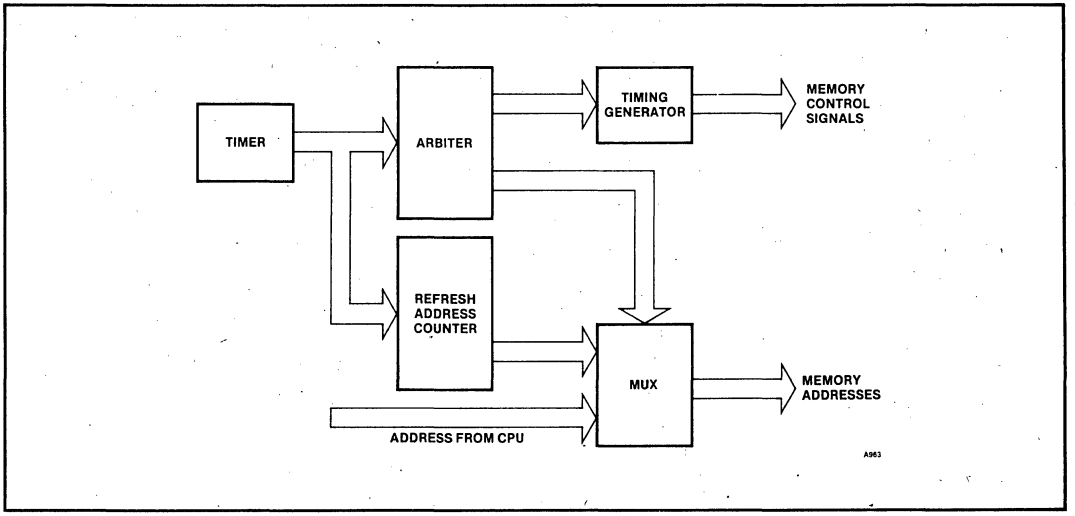

Figure 1. Memory Control Block Diagram

Figure 2 shows a simple microprocessor memory system implemented with three major blocks: the CPU, the memory array, and a memory controller. An example of this configuration is a system comprising an 8088 CPU, and 8203 DRAM controller and a 2164A memory array. To advance this configuration to a higher level of integration would require a decision on whether to place the memory control inside the CPU or within the memory itself.

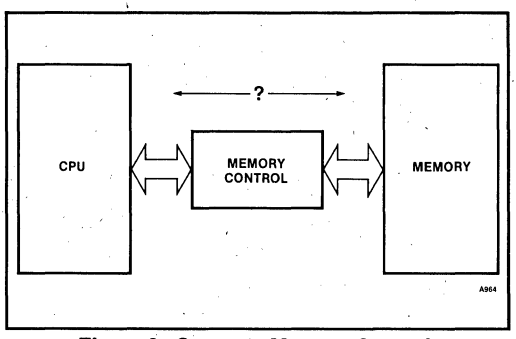

Figure 2. Separate Memory Control

Memory control incorporated within the CPU requires CPU participation in all memory references — just to preserve refresh. This includes DMA (direct memory access) which normally doesn't require or permit CPU intervention. Also, the CPU must run continuously. Single stepping, hold operations, extended WAIT states and the special block data move instructions of some microprocessors must all be carefully avoided to preserve refresh and maintain data integrity of the memory system. While these constraints can be acommodated with careful design, the added overhead does limit the full CPU processing capabilities and overall system performance.

A sensible alternative is to integrate the memory controller circuits into the memory — completely freeing the CPU of this task. While this approach places an additional burden on the device designer, it greatly simplifies the task of the system designer by eliminating the design problems associtated with refresh and timing. This permits a very simple interface to the CPU and yet provides guaranteed refresh, optimized timing, and minimal hardware support requirements.

A microprocessor integrates all the components of a central processing unit into one device. An iRAM integrates all the components of a dynamic RAM memory system into a single device. This is unlike the pseudostatic or quasi-static RAM devices which only incorporate a portion of the refresh circuitry onto the memory chip and still require much control from the CPU. The integration used in the iRAM includes the refresh timer, refresh address control and counter, address multiplexing, and memory cycle arbitration as well as an 8-bit wide memory array. Figure 3 is a pictorial representation of this concept.

1.3 Memory System Size and Cost Constraints

Integrated RAMs are primarily intended for use in microprocessor memories usually less than or approximately equal to 64K bytes, while standard DRAMs with a separate controller are more cost effective in larger memories. The relative costs of systems designed with various device family types are shown in Figure 4. A range is shown for each alternative to represent the change in cost over time. Thus, the 2K × 8 SRAM is a good choice for very small memory systems of less than 8K bytes while DRAMs provide a clear advantage in the region beyond 64K bytes. In the region between 8K and

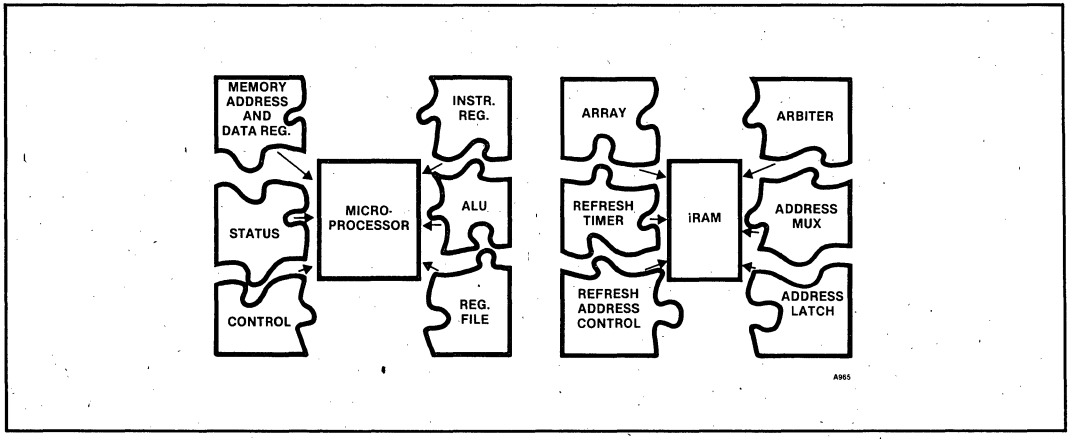

Figure 3. iRAM/Microprocessor Comparison

64K, however, standard DRAMs are usualy not as cost effective because of the overhead involved in the design and cost of the hardware for the controller. Based on these comparisons, iRAMs have a clear advantage for anything other than very small or very large memory systems.

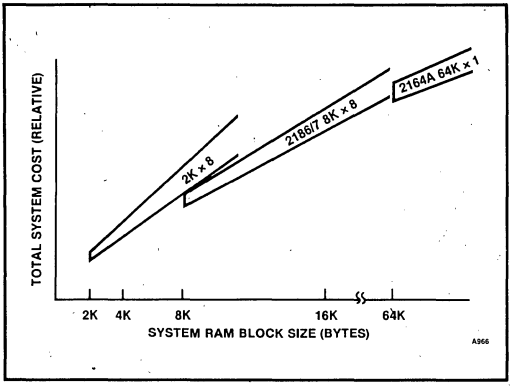

Figure 4. System Cost Graph

1.4 Byte-wide Universal Memory Site

The byte-wide universal memory site concept allows a system designer to create one or more memory sites that can accommodate several types of x8 memories, including RAMs, ROMs, EPROMs, and E^2PROMs. The universal site is depicted in Figure 5. Though based on a 28-pin site, the universal site also supports 24-pin devices. For this site to be truly universal, it should contain provisions for memory densities that have not yet been developed.

Figure 6 shows various memory classes and how they conform to the universal site. The universal site is particularly useful in development of microprocessor systems in which the hardware design of the memory site may be completed early in the design cycle before the RAM/ROM mix has been specified. For example, a RAM might be initially used to store microprocessor instruction code during the development and testing of the system software. This allows code to be run and debugged at full system speed. Initial prototypes and small production runs can place EPROMs in the same sockets, while full scale production may change to PROMs or ROMs. The universal site flexibility also allows an easy upgrade path to next generation (higher density) devices.

A key feature of the universal memory site is the two-line bus control with separate $\overline{CE}$ and $\overline{OE}$ to prevent bus contention in a system. This convention offers a distinct advantage over devices with only one-line control. (Eliminating the effects of bus contention is extremely important and not always easy due to its subleties. Generally, the current and voltage spiking on the power supply rails presents the major problem because this type of noise can lead to a whole host of problems including invalid data, false triggering, race conditions, and reflections, to name a few.)

1.4.1 ONE-LINE CONTROL

With one-line control devices (Figure 7), bus contention occurs when two devices simultaneously occupy a bus (when $\overline{CE}$ of one device goes inactive simultaneously with another devices' $\overline{CE}$ going active). This is the usual situation when chip selects are generated from a decoder. The contention occurs because it takes more time for the output of the deselected device to turn off (switch to high impedence) than the short output buffer turn-on time of the selected device. Because the data lines are wire-ORed to a common data bus, any data bits of opposite polarity will cause bus contention (Figure 8).

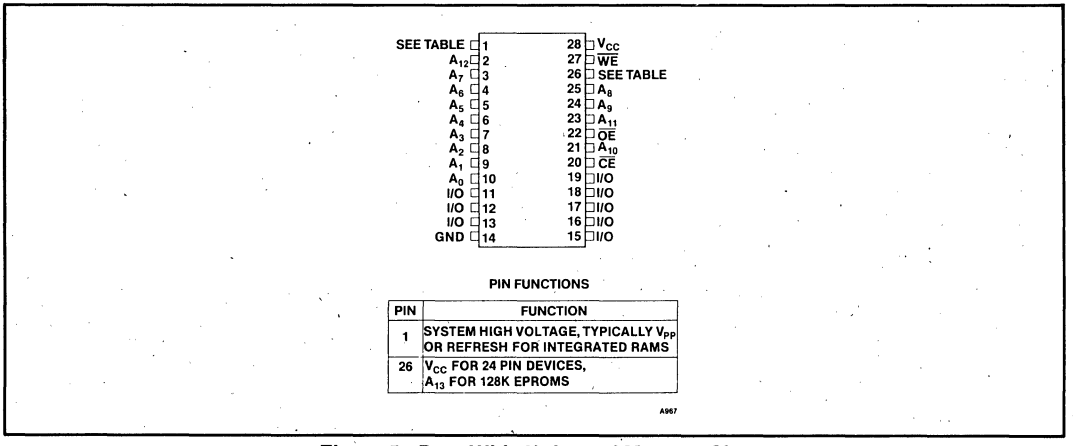

Figure 5. Byte-Wide Universal Memory Site

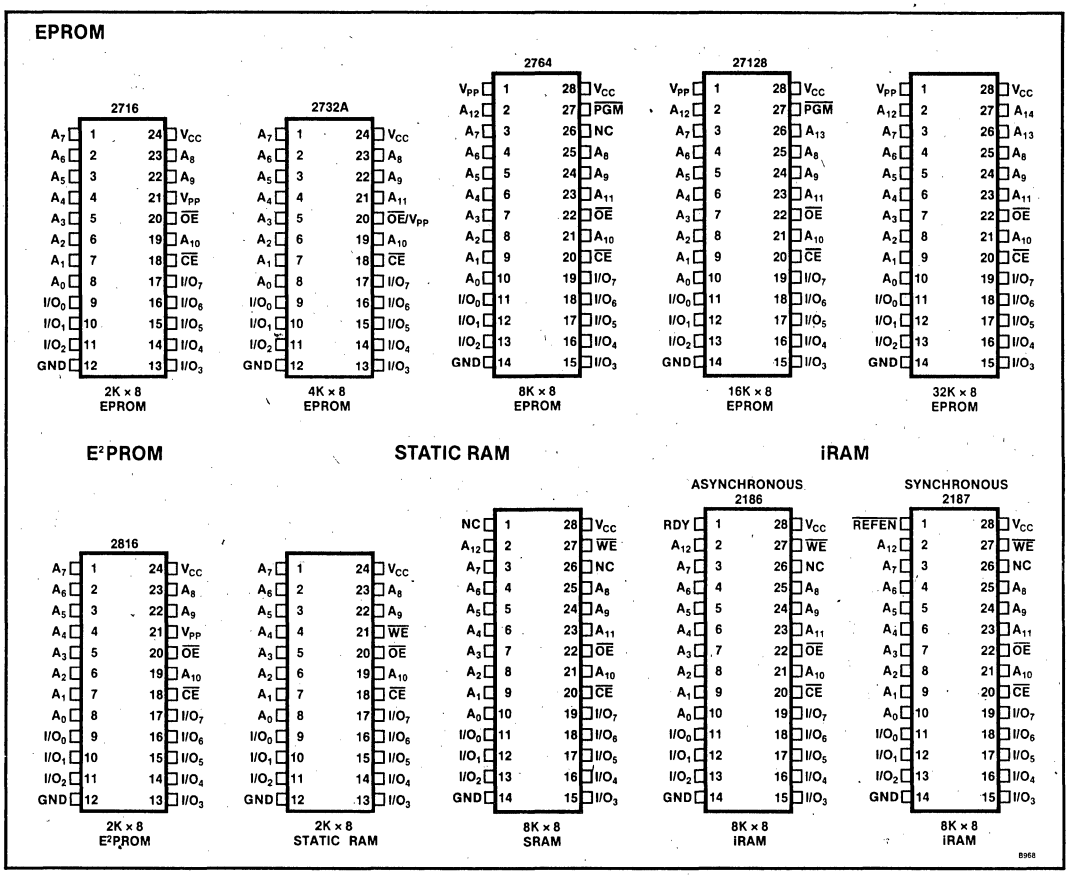

Figure 6. Intel's Line of Universal Products

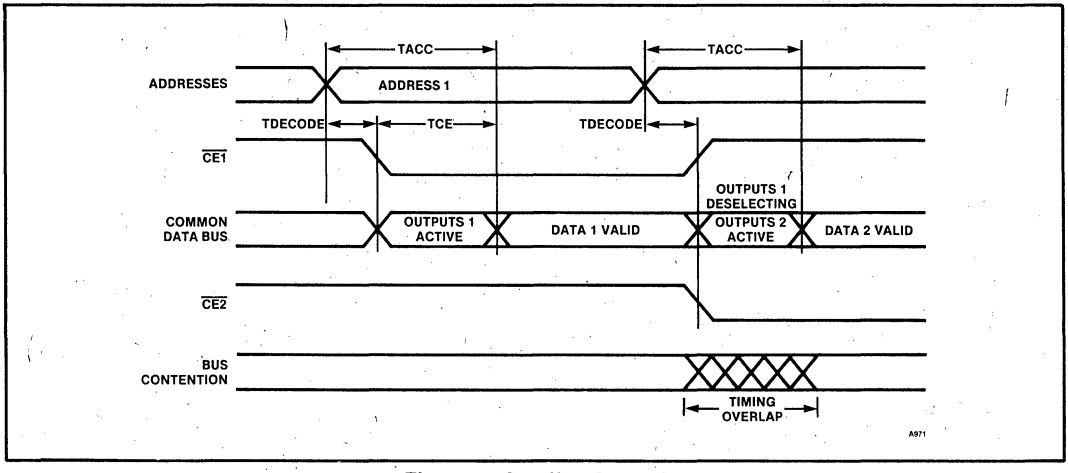

Figure 7. One-line Control

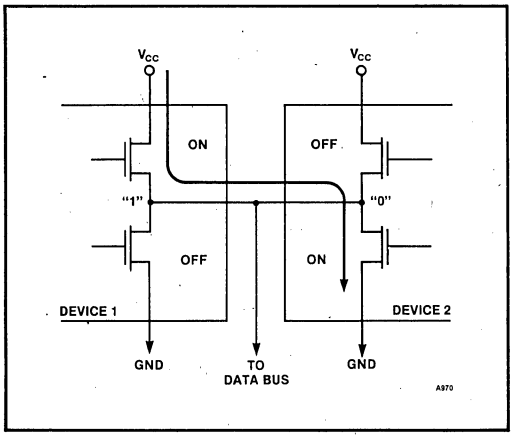

Figure 8. Bus Contention

1.4.2 TWO-LINE CONTROL

Similar to one-line control, two-line control logic allows the $\overline{CE}$ of one device to go inactive simultaneously with another going active. However, the timing diagram in Figure 9 shows that no bus contention occurs because the $\overline{OE}$ of the selected device is not enabled until the outputs of the deselected device have switched off the bus.

The use of an independent output enable is the best way to eliminate bus contention in the system. The use of non-integrated output buffers cannot achieve the same result; they can only confine bus contention to a memory card or memory section of a large card. In addition, as processor speeds increase, greater demands are placed on memory performance and the use of external non-integrated output buffers places still more constraints on memory system performance. In this context, the time between addresses out and data in is a fixed interval for any given processor. All devices inserted in the path, demultiplexers, transceivers, decoders, etc., must be compensated for by a higher speed memory.

2 DEVICE DESCRIPTION

2.1 Overview

The 2186 and 2187 iRAMs are 5-volt only, dynamic RAM 8K × 8 systems integrated on a single chip (Figure 10). The memory devices have been designed for easy use with microcontrollers, multiplexed address/data bus microprocessors, and processors with separate address and data paths. These memories are referred to as integrated RAMs or "iRAMs" because they contain refresh timing and control logic. The 2186/87 iRAMs include the following major features:

- Easy to use on-chip self-refresh, including:
  - — Internal refresh timer
  - — Refresh address counter
  - — High speed arbiter (2186 only)
  - — Refresh address multiplexer
  - — Complete internal timing control
- External refresh control option (2187 only)
- Microprocessor handshake signal (2186 only)
- Outputs drive two low power Schottky TTL loads and 100 pF

The 2186/87 iRAMs are fabricated using an N-channel double layer polysilicon gate process with depletion loads. The four-quadrant memory array is built with conventional one transistor DRAM cells, polysilicon word lines and folded metal bit lines. Each of the four quadrants contains 128 rows and columns. In addition, four redundant columns and four redundant rows are provided. Two pairs of I/O lines from each of the quad-

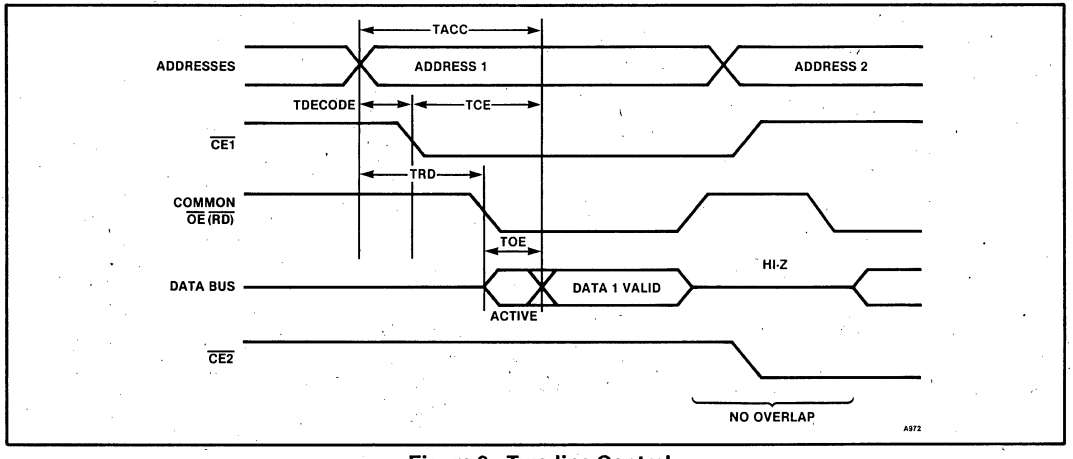

Figure 9. Two-line Control

rants provide a total of eight bits to the data bus. An active restore circuit boosts the bit lines back to a full V_{CC} level after every read or refresh cycle. Boosted word lines and column select lines are used to write a full V_{CC} level into the memory cells. Wide internal operating margins provide a high degree of reliability.

2.2 Device Pinout

The pinout of the 2186 and 2187 is shown in Figure 11. The industry standard 28-pin package conforms to Intel's byte-wide universal memory site (Section 1.4). Pin 1 (labeled "CNTRL") is the only external difference between the 2186 and 2187. On the 2186, Pin 1 is a RDY output — a signal to the system indicating memory status. Pin 1 on the 2187 is a "refresh" strobe ($\overline{\text{REFEN}}$), an input signal for external refresh requests.

Pins 2 thru 10, 21, and 23 thru 25 are the 12 address inputs required to select each of the 8192 bytes. Pins 11 through 13 and 15 through 19 are the eight bits of the bidirectional data bus.

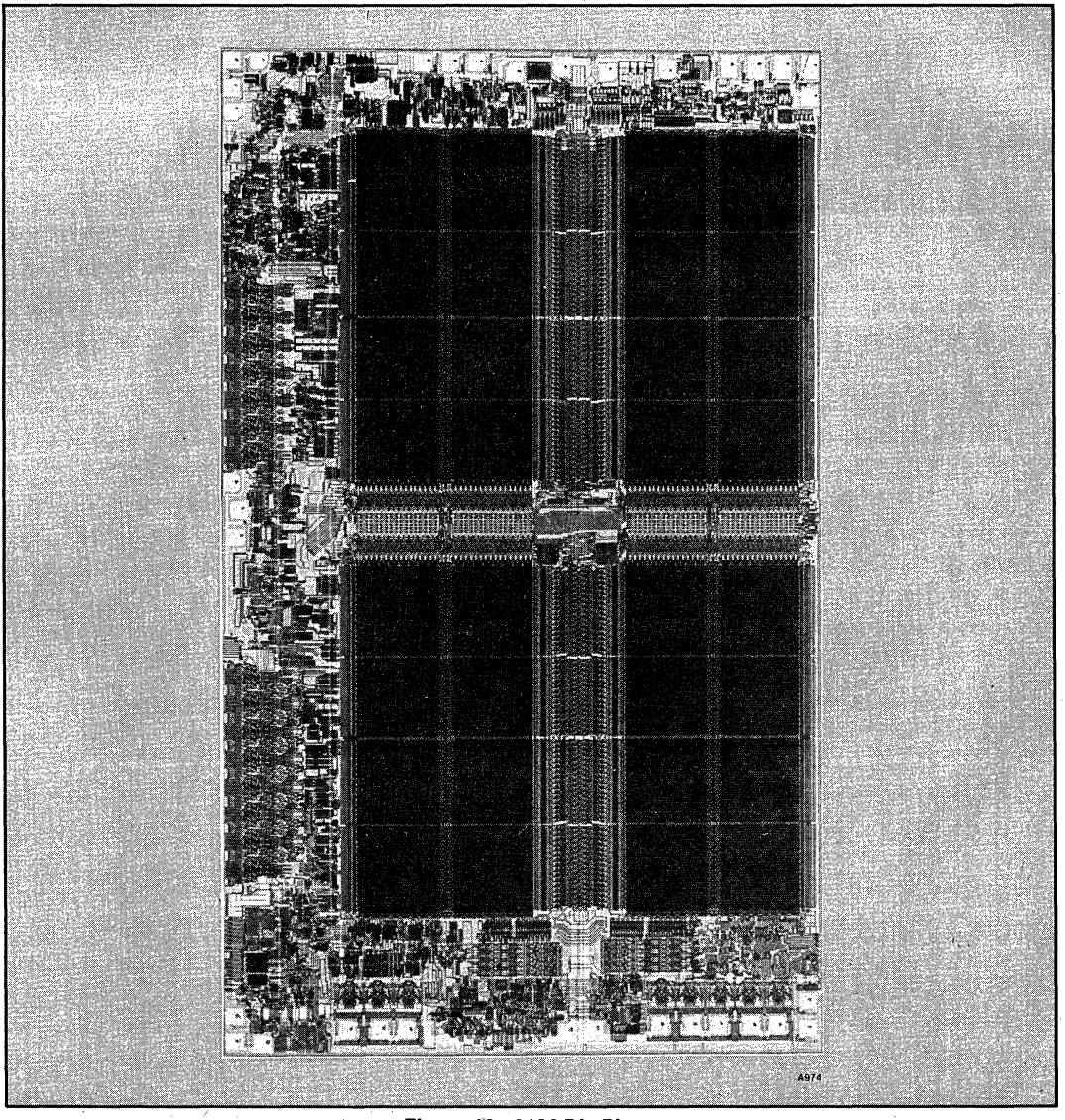

Figure 10. 2186 Die Photo

Pin 27 is the write pulse input strobe ($\overline{WE}$) for data store during Write cycles. Pin 20 is Chip Enable ($\overline{CE}$), which latches addresses and begins the internal memory cycle. Pin 22 is Output Enable ($\overline{OE}$), normally connected to a CPU READ ($\overline{RD}$) line. $\overline{OE}$ enables the iRAM output buffers during a Read cycle.

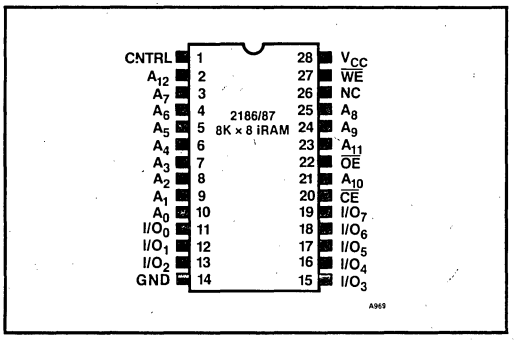

Figure 11. 2186/87 Pinout

2.3 Internal Description

2.3.1 ASYNCHRONOUS AND SYNCHRONOUS REFRESH

The 2186 iRAM contains automatic internal refresh circuitry making it an ideal choice for asynchronous applications. The 2187 does not have the internal arbitration capability as it has been designed for use in synchronous applications.

Pin 1 on the 2186 is the RDY output which serves as the handshake signal (required in asynchronous systems) and is usually bussed to the RDY input circuit of the processor. The RDY output is an open drain device, requiring a 510 ohm pull-up resistor which allows "wire-OR" connections of other device RDY outputs without the need for extra gates.

The 2187 receives external refresh requests via Pin 1 ($\overline{REFEN}$). This input must be strobed 128 times within 2 milliseconds to perserve refresh in the dynamic RAM array. The 2187 iRAM is designed for use in synchronous systems where the user wants control of the refresh cycles. Hence, the designer must provide refresh requests to the iRAM. The 2187 has neither a RDY signal nor any access cycle deferment and because it has no built-in arbitration capabilities, the user must also guarantee that access cycles are not requested during refresh cycles.

Refresh addresses are generated internally in both devices by an onboard refresh address counter. In addition, both devices have an internal refresh timer which, for the 2187, becomes active in a power-down mode.

2.3.2 FUNCTIONAL BLOCK DIAGRAM

Figure 12 shows a functional block diagram of the iRAM.

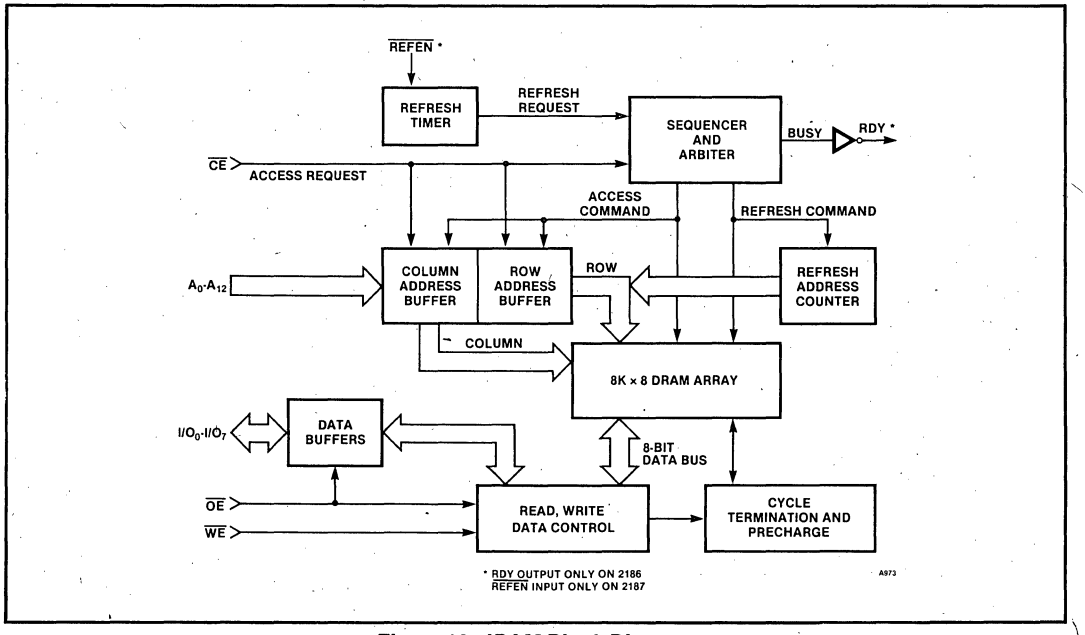

Figure 12. iRAM Block Diagram

2.3.2.1 Refresh Timer

The refresh timer requests refresh cycles as required. The refresh timer has been designed to track with temperature and process variations. The design optimizes the rate at which refreshes occur while still guaranteeing data integrity.

2.3.2.2 Sequencer and Arbiter Circuits

The sequencer and arbiter circuits accept refresh requests from the refresh timer and memory cycle requests from the $\overline{CE}$ input. The internal refresh command and the external memory accesses are asynchronous and either may occur at any time with respect to the other. If one does occur while the other is in progress, the request is queued and the cycle performed after the existing cycle has completed. If a refresh cycle is already in progress at the time an access request occurs, the RDY signal on pin 1 is pulled to V_{OL} informing the system that the access cycle is being deferred. In this instance, the normal cycle will be delayed until after the refresh cycle has been completed. RDY will remain low until shortly before valid data becomes available, after which the cycle is completed in a normal manner. The internal high speed arbiter resolves any conflict wherein an internal refresh command and an external access occur simultaneously. This circuit also generates the RDY handshake signal in the 2186. The sequencer/arbiter circuit also decides which type of memory cycle is to occur and controls the operation.

2.3.2.3 Address Buffers and Refresh Address Counter

External addresses A_0-A_{12} are directed to internal row and column address buffers to generate internal byte addresses. Refresh addresses are generated by an internal refresh address counter and are multiplexed internally with the external row addresses.

2.3.2.4 Data Buffers

Controlled by signals from the read/write data control circuit, the three-state bidirectional data buffers receive or transmit eight data bits.

2.3.2.5 Read, Write Data Control

The read/write data control circuit controls and directs the flow of data between the 8K × 8 DRAM memory array and the data buffers.

2.3.2.6 Cycle Terminator and Precharge

The cycle terminator and precharge circuits ensure proper termination of all memory cycles and precharge the dynamic circuitry in preparation for the next cycle.

3 DEVICE DESCRIPTION

All timing signals used throughout this document are denoted by various alpha character strings to indicate certain basic conditions or parameters. Understanding signal name derivation will enable the reader to arrive at a correct interpretation of any signal name encountered. Figure 13 illustrates the meaning of various letters used in a signal name.

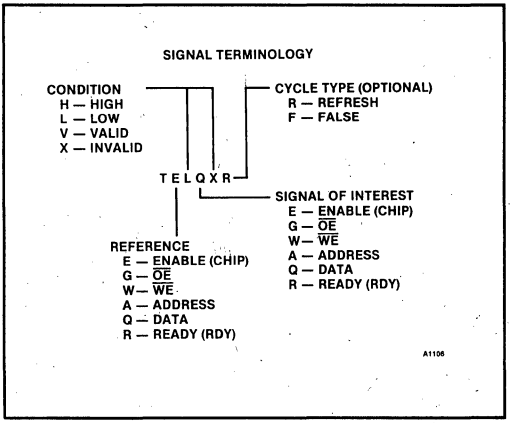

Figure 13. Timing Signal Terminology

Each control signal is given a one-letter designer; i.e., $\overline{CE}$ is represented by E, $\overline{OE}$ by G, etc. Each of these letters is followed by another letter describing the state of the foregoing. In addition, a timing descriptor may have a letter added to the end to describe a special case. For example, TELGL is the time from $\overline{CE}$ low to $\overline{OE}$ low, while TEHELF is the time from $\overline{CE}$ high to the next $\overline{CE}$ low during a false memory cycle; TELQVR is the time from $\overline{CE}$ low to data valid for a not ready condition.

The 2186 and 2187 are edge-triggered devices that recognize a timing edge as a signal to start an operation. Because of this, $\overline{CE}$ must be allowed to make only one transition per cycle, otherwise the device cycle time (TELEL) will be violated. The 2186 and 2187 latch all external addresses on the leading edge of $\overline{CE}$. Data is latched into the device on the **leading** edge of $\overline{WE}$ as opposed to the trailing edge write requirement which is common among static RAMs.

The 2186 provides four major types of cycles: read, write, false memory, and refresh.

Two major modes of operation exist for both read and write cycles; $\overline{CE}$ pulsed mode and $\overline{CE}$ long mode. For pulsed mode $\overline{CE}$ operation, the low $\overline{CE}$ time (TELEH) must be less than or equal to $TELGL(TELWL)_{max} + TGLEH(TWLEH)_{min}$, while long $\overline{CE}$ mode requires a longer $\overline{CE}$. (For more detailed timing information, consult the 2186 and 2187 data sheets.)

3.1 Read Cycle

A read cycle (Figure 14) is initiated by both $\overline{CE}$ and $\overline{OE}$ going low during the same cycle. Depending on the low time of $\overline{CE}$, either a pulsed or long $\overline{CE}$ mode will occur.

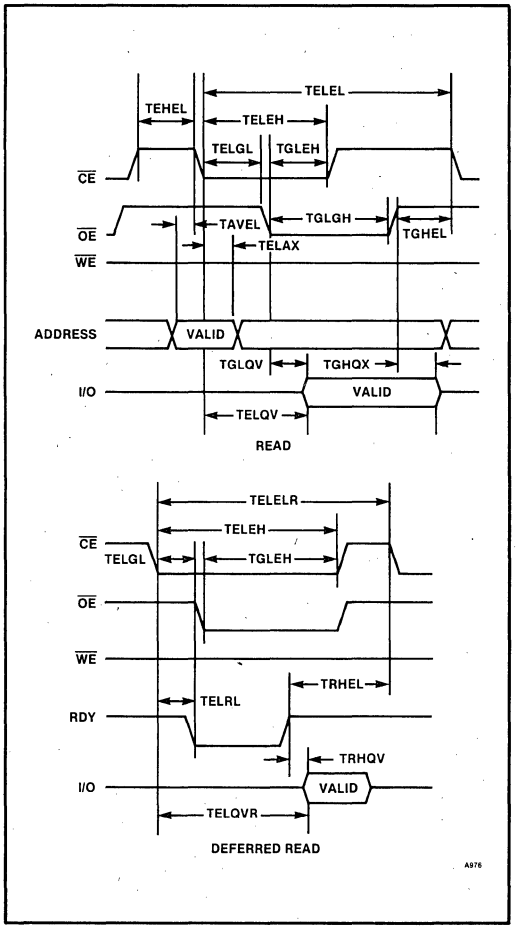

Figure 14. Read Cycle Timing

3.1.1 PULSED MODE $\overline{CE}$ READ

For pulsed mode, a $\overline{CE}$ read cycle is initiated on the falling edge of $\overline{CE}$ at which time either a refresh is or is not in progress.

Refresh cycle not in progress

With a refresh cycle not in progress, the memory cycle can immediately commence (non-deferred read cycle). After the falling edge of $\overline{CE}$, $\overline{OE}$ must go low within a specified period of time (TELGL). If this latter condition is not met, a false memory cycle (FMC) will occur (see Section 3.3). At some point after $\overline{OE}$ goes low, data will become valid and remain so for as long as $\overline{OE}$ is active, independent of $\overline{CE}$.

Refresh cycle is in progress

If a refresh cycle is in progress at the time $\overline{CE}$ goes low, the read cycle will be delayed (deferred read cycle) until after the refresh cycle has completed. In this event, the 2186 will respond very quickly with a RDY low output (TELRL). After the refresh cycle is completed, the read cycle will commence and data will be available at a given time after RDY returns high (TRHQV). As was the case with the non-deferred read cycle, TELGL must be met or an FMC will occur.

3.1.2 LONG $\overline{CE}$ MODE READ

For long $\overline{CE}$, a read cycle mode is initiated on the falling edge of $\overline{CE}$. Similarly to pulsed mode $\overline{CE}$, both deferred and non-deferred write cycles may occur where a deferred cycle causes RDY to be pulled low.

In the long $\overline{CE}$ mode of operation, $\overline{CE}$ must be held low for a given period of time after $\overline{OE}$ goes low (TGLEH). Violation of this specification will cause an FMC to occur. At a given time after $\overline{OE}$ goes low, valid data will become and remain available throughout the duration of $\overline{OE}$'s active period, independent of $\overline{CE}$.

Note that deferred access cycles are not allowed for the 2187.

3.2 Write Cycle

A write cycle (Figure 15) occurs when both $\overline{CE}$ and $\overline{WE}$ go low during the same cycle. As is the case for the read cycle, either a pulsed or a long $\overline{CE}$ mode can occur.

3.2.1 PULSED MODE $\overline{CE}$ WRITE

In the pulsed mode, a $\overline{CE}$ write cycle is initiated on the falling edge of $\overline{CE}$. At this time, a refresh cycle may or may not be in progress.

Refresh cycle not in progress

With a refresh cycle not in progress, the memory cycle can immediately commence (non-deferred write cycle). After the falling edge of $\overline{CE}$, $\overline{WE}$ must go low within a specified period of time. If this latter condition is not met, a false memory cycle (FMC) will occur (see Section 3.3). On the falling edge of $\overline{WE}$, data is latched into the device.

Refresh cycle is in progress

If a refresh cycle is in progress at the time $\overline{CE}$ goes low, the write cycle will be delayed (deferred write cycle) until after the refresh cycle has completed. In this event, RDY is brought low and held there until the refresh cycle has completed. Note that data is still latched into the 2186 on the falling edge of $\overline{WE}$.

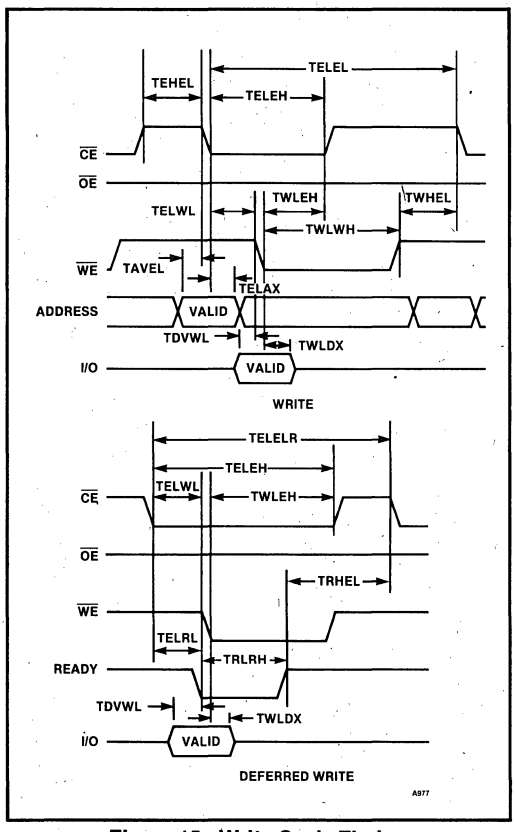

Figure 15. Write Cycle Timing

3.2.2 LONG $\overline{CE}$ MODE WRITE

A long mode $\overline{CE}$ write cycle is initiated on the falling edge of $\overline{CE}$. As is the case for a pulsed mode $\overline{CE}$, both deferred and non-deferred write cycles may occur with RDY being pulled low in the deferred cycle.

For the long $\overline{CE}$ mode of operation, $\overline{CE}$ must be held low for a given period of time after $\overline{WE}$ goes low (TWLEH). Violation of this specification will cause an FMC to occur. On the falling edge of $\overline{WE}$, data is latched into the device.

3.3 False Memory Cycle (FMC)

A false memory cycle (Figure 16) occurs when $\overline{CE}$ is active and neither $\overline{OE}$ or $\overline{WE}$ go low. In this case, the cycle will automatically be terminated on the trailing edge of $\overline{CE}$. This is a valid mode of operation in which precharge and data integrity are guaranteed.

As an added feature of the false memory cycle, a refresh cycle is performed on the row which is selected by the seven external row addresses.

Note that the $\overline{CE}$ high time (TEHELF) required after an FMC is somewhat longer than the corresponding period required for a read or write cycle (TEHEL).

As is the case with a read or write cycle, FMC cycles can be deferred. RDY response time (TELRL) and recovery time (TRHEL) are the same as for the read and write cycles.

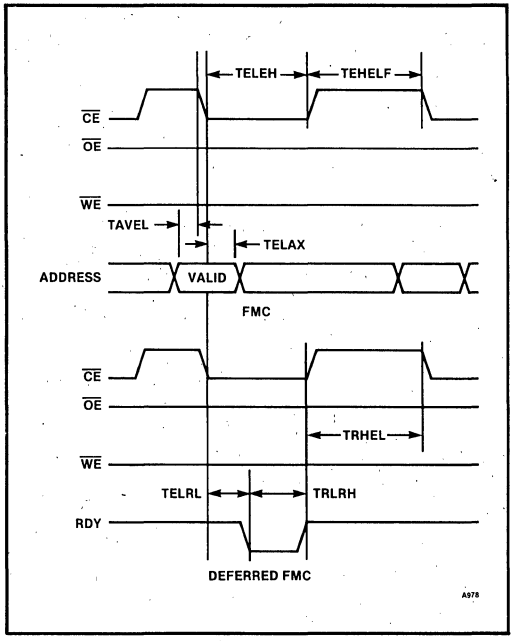

Figure 16. False Memory Cycle Timing

3.4 Refresh Modes

Both the 2186 and 2187 can be refreshed by reading or writing all 128 rows (A_0 through A_6) within a two millisecond period. Several specific modes of refresh operation exist for each part as outlined below.

3.4.1 2186 AUTOMATIC INTERNAL REFRESH

Refresh is totally automatic and requires no external control. In addition, the refresh address is computed internally and does not have to be supplied externally. A high speed arbitration circuit resolves any potential conflict arising between simultaneous access and refresh cycle requests. If a refresh cycle is in progress at the time $\overline{CE}$ becomes active (low), the 2186 will respond with a RDY low output. If, on the other hand, an access or false memory cycle is in progress at the time the internal refresh timer times out, the refresh request will be queued and then performed after the present cycle is complete. Note that RDY will not go low during a refresh unless the RAM is selected by $\overline{CE}$.

3.4.2 2187 EXTERNAL REFRESH

A high-to-low transition on the $\overline{\text{REFEN}}$ input will cause a refresh cycle to be initiated (Figure 17). In this mode $\overline{\text{REFEN}}$ must always be strobed 128 times in a two millisecond period. The $\overline{\text{REFEN}}$ input may be strobed in distributed or burst mode. Refresh addresses are supplied by an internal refresh address counter.

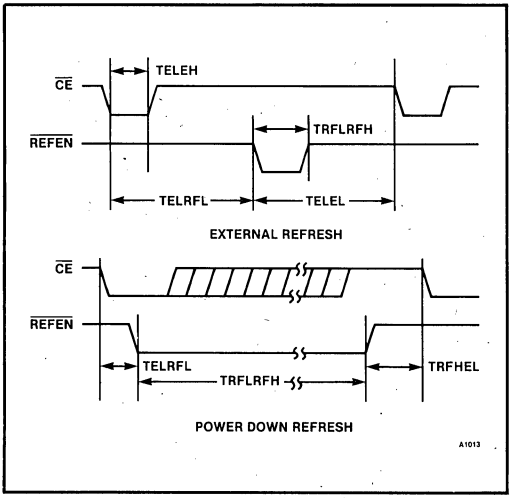

Figure 17. External Refresh Timing

3.4.3 2187 POWER-DOWN AUTOMATIC INTERNAL REFRESH

If $\overline{\text{REFEN}}$ is kept low for greater than one timer period, the internal refresh timer will be activated. Refresh in this mode is totally automatic and requires no external stimulus. $\overline{\text{REFEN}}$ must return high within a specified interval prior to the next memory access cycle (TRFHEL). Attempting to access the 2187 during a refresh cycle is not a valid mode of operation.

3.5 Single-step Operation

Both the 2186 and 2187 can support "single-step" operation for microprocessor system diagnostics. Single-step operation is defined as inserting an unspecified number of WAIT states in the middle of a normal RAM access.

3.5.1 2186

The 2186 supports single-step operation in microprocessor applications which hold $\overline{\text{OE}}$ or $\overline{\text{WE}}$ valid (low) for indefinite periods. Data will remain valid on the bus as long as $\overline{\text{OE}}$ is valid. $\overline{\text{WE}}$ latches data on its falling edge. Automatic refreshes will continue to be performed as needed, even while $\overline{\text{OE}}$ or $\overline{\text{WE}}$ is held low. During this extended cycle, the internal array is free to be refreshed with no threat of access/refresh cycle conflicts. Because of this, RDY will not respond to these extended cycle refreshes.

3.5.2 2187

The 2187 supports single-step operation by following the beginning of a memory cycle with $\overline{\text{REFEN}}$ going and remaining low. Refresh cycles continue to occur periodically as long as $\overline{\text{REFEN}}$ is held low, even if $\overline{\text{OE}}$ or $\overline{\text{WE}}$ remain low indefinitely. Data remains valid on the bus as long as $\overline{\text{OE}}$ is valid. $\overline{\text{WE}}$ latches data on its falling edge. Again, after $\overline{\text{REFEN}}$ returns to a high state, a minimum amount of time (TRFHEL) must be allowed before the next high-to-low transition of $\overline{\text{CE}}$.

3.6 Power-up

3.6.1 2186

To guarantee power-up, all control inputs must be inactive (high) for a 100 microsecond period after V_{CC} is within specification. No dummy cycles are required.

3.6.2 2187

The 2187 power-up is accomplished by holding REFEN active low for 100 microseconds after V_{CC} is within specification. All inputs must be stable and within specification. $\overline{\text{CE}}$, $\overline{\text{WE}}$, and $\overline{\text{OE}}$ must remain inactive (high) during user power-up.

4 INTERFACE CIRCUITRY

There are three key interface circuit considerations when designing with iRAMs.

1. The first consideration is the need for a single edge ("glitchless") transition of chip enable ($\overline{\text{CE}}$) per cycle — because the leading edge transition (active low) of $\overline{\text{CE}}$ latches addresses into the iRAMs and initiates several internal device clocks. Also, there is a minimum specification for $\overline{\text{CE}}$ inactive time (to allow for proper precharge of internal dynamic circuitry).
2. The second consideration concerns write cycles. Because iRAMs write data on the leading edge of $\overline{\text{WE}}$, there is the need for valid data at the memory device before the $\overline{\text{WE}}$ line is activated.
3. The third consideration is the value of same site compatibility with byte-wide SRAMs, EPROMs, ROMs, and E^2PROMs. In particular, allowance for the trailing edge write requirements of SRAMs should be made.

Modest additional circuitry permits compatibility with SRAMs as second sources or allows the iRAM to substitute for ROM or EPROM during debug stages. Several

circuit examples to meet the various requirements of interfacing microprocessors to iRAMs will be described.

Figure 18 shows circuitry for generation of a "glitchless" $\overline{CE}$ from standard 8086 bus signals. Figure 19 shows the circuit timing. This dual J-K flip-flop arrangement guarantees a number of operating conditions. The flip-flops generate a stable $\overline{CE}$ for the iRAMs by enabling the 8205 decoder only after valid addresses have arrived, but early enough to allow the 2186 iRAMs' RDY signal to respond in time to insert a WAIT-state (if required). The circuit also ensures that a minimum $\overline{CE}$ high time is provided. (This is especially important during false memory cycles (FMC) where the $\overline{CE}$ high time specification stretches beyond that of normal cycles).

Also of significance is the compatibility of this circuitry with SRAMs and EPROMs. This includes requiring $\overline{CE}$ to remain valid throughout the cycle.

The interface circuit is simply a two-bit counter designed to start a count sequence when flip-flop A is preset by ALE going high. The $\overline{Q}$ output of flip-flop B along with M/$\overline{IO}$ ($\overline{S2}$ for max mode) is used to enable the $\overline{CE}$ decoder to provide a $\overline{CE}$ to the desired iRAM.

The READY signal is ANDed with the $\overline{Q}$ output of flip-flop A and input into flip-flop B. As long as READY is low, the K input of the flip-flop driving B will stay low, keeping it from being reset. This in turn acts to keep $\overline{CE}$ active. This input allows $\overline{CE}$ to stretch during a WAIT state to meet the requirements or SRAMs or EPROMs that may occupy the same memory site. However, the iRAMs do not require that $\overline{CE}$ be held low for extended cycles.

The circuit in Figure 20 (only for the 8088 — enclosed in dashed lines) offers an alternative. This circuit provides an Enable signal ($\overline{E}$) for the $\overline{CE}$ decoder which is synchronized with ALE. This Enable signal along with

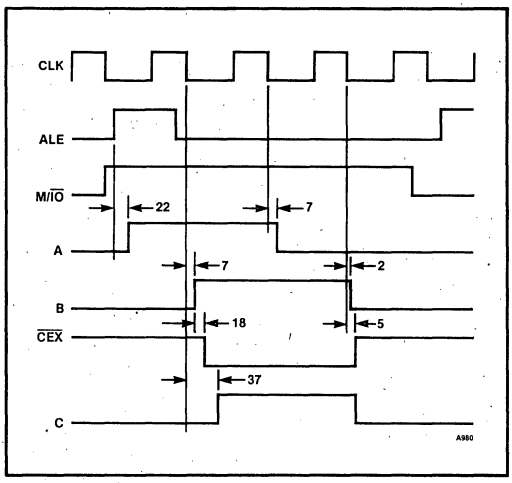

Figure 19. Interface Circuit No. 1 Timing

M/$\overline{IO}$ ($\overline{S2}$ for max mode), is used to enable and address decoder to provide $\overline{CE}$'s to the iRAMs.

A certain amount of skew can occur between the falling edge of ALE and the falling edge of the clock. Two situations can occur: (1) ALE goes low before the falling edge of the clock, the $\overline{E}$ enable line to the decoder remains high until the falling edge of the clock, and (2) ALE goes low at or after the falling edge of the clock, in which case the $\overline{E}$ enable line is immediately activated and enables the decoder. Note that the $\overline{RESET}$ line is used to clear the M/$\overline{IO}$ flip-flop. This causes the 74S138 to be disabled, satisfying the power-up requirements of the 2186 ($\overline{CE}$ remains high). Also, a pull-up resistor is connected to the $\overline{RD}$ line. This ensures that $\overline{OE}$ remains high during RESET (the 8086 three-states $\overline{RD}$ during RESET).

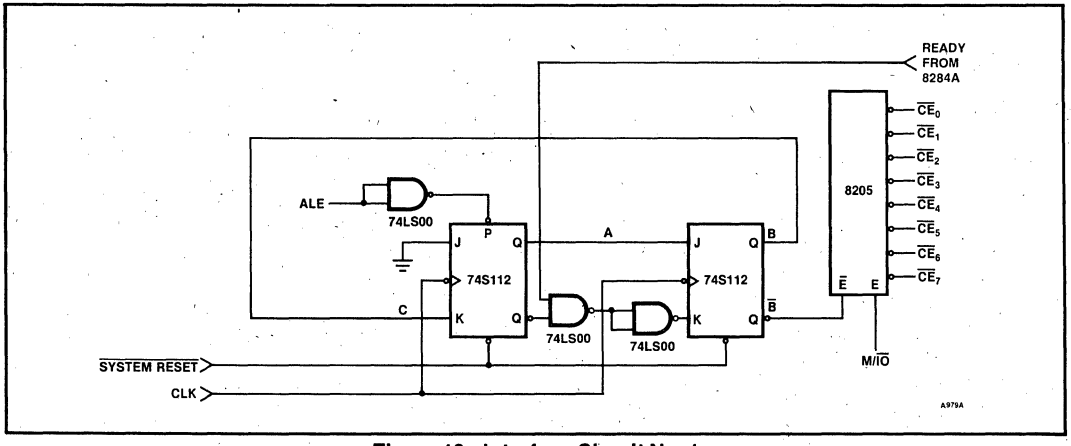

Figure 18. Interface Circuit No. 1

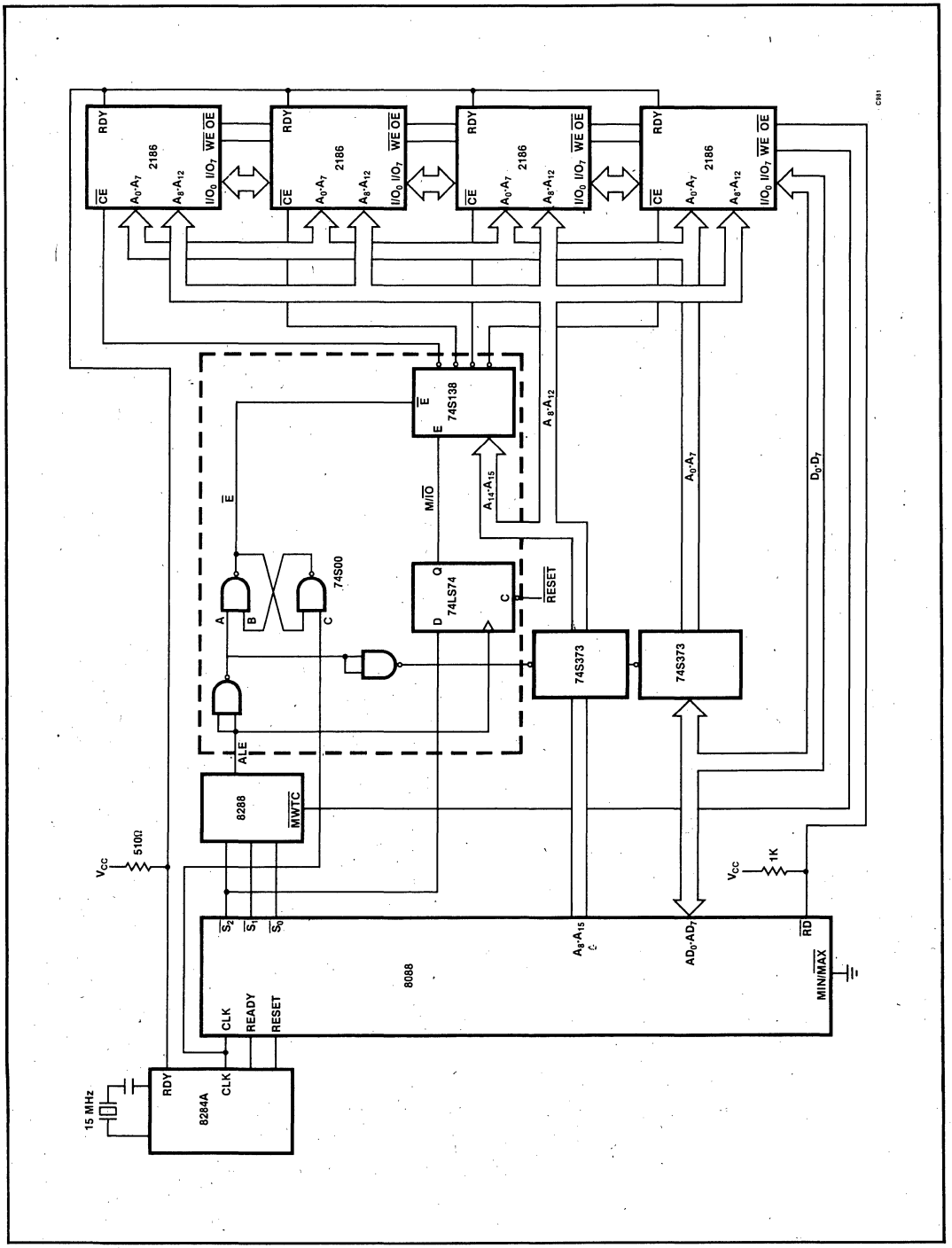

Figure 20. Interface Circuit No. 2

Either circuit will provide all of the interface needed for a 5 MHz 8086 or 8088 max mode system, because MWTC can be used to provide both leading and trailing edge writes. For a min mode system, the circuit in Figure 21 can be used to provide a leading edge write, and the circuit in Figure 22 can be used to provide both a leading and trailing edge write.

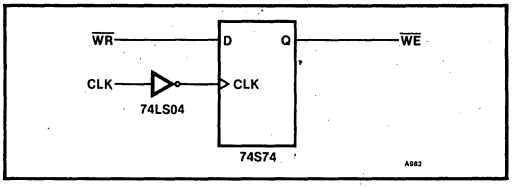

Figure 21. Leading Edge Write

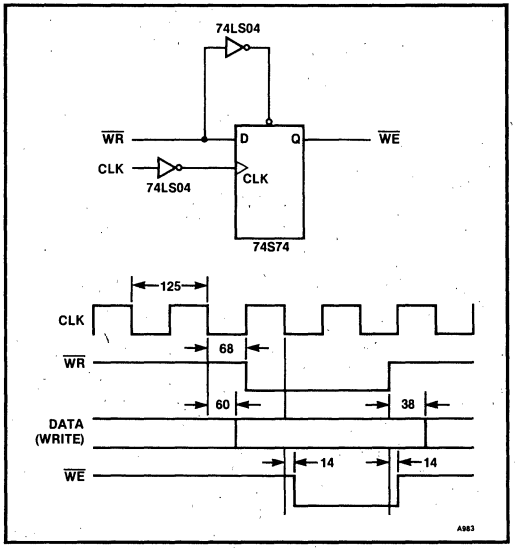

Figure 22. Leading and Trailing Edge Write

A simple, one-gate alternative to the preceding example along with the appropriate timings is shown in Figure 23. This cross-coupled NAND arrangement operates in much the same way as the $\overline{CE}$ generation circuit presented earlier, acting to synchronize the $\overline{WR}$ pulse with the clock. This circuit will provide for both leading and trailing edge writes.

5 SPECIFIC APPLICATION EXAMPLES

This section describes some typical memory interface designs using three types of CPUs: an 8-bit microcontroller, an 8-bit microprocessor, and a 16-bit microprocessor. Design examples are included for both the 2186 and the 2187.

5.1 8-Bit Microcontroller

Figure 24 shows a two-chip microcomputer system using the 8751/8051. This system features 4K bytes of EPROM/PROM and 8K bytes of data storage using the 2186 iRAM. Interface to the multiplexed bus is simplified because the 2186 latches addresses from its external bus on the falling edge of $\overline{CE}$, eliminating the need for latches. In this configuration, the ALE output from the microcontroller is gated with P2.7, and used to generate $\overline{CE}$ of the 2186. The gating of ALE with P2.7 is important for the following reasons: when the 8051 does any type of memory operation, it outputs ALE onto its external bus. This includes internal program memory fetches, in which the ALE cycle time (Figure 25) is only half of what it would be for an external data memory fetch. During these "short" cycles, ALE must be inhibited from generating a $\overline{CE}$ to the 2186, or else the 2186 cycle time with WAIT specification (TELELR) would be violated. To carry this out, P2.7 is initially set to a "1" which is done automatically upon RESET. This "1" will be present on the output during all times except external data memory fetches from addresses below 8000H, at which time P2.7 will go low, allowing ALE to

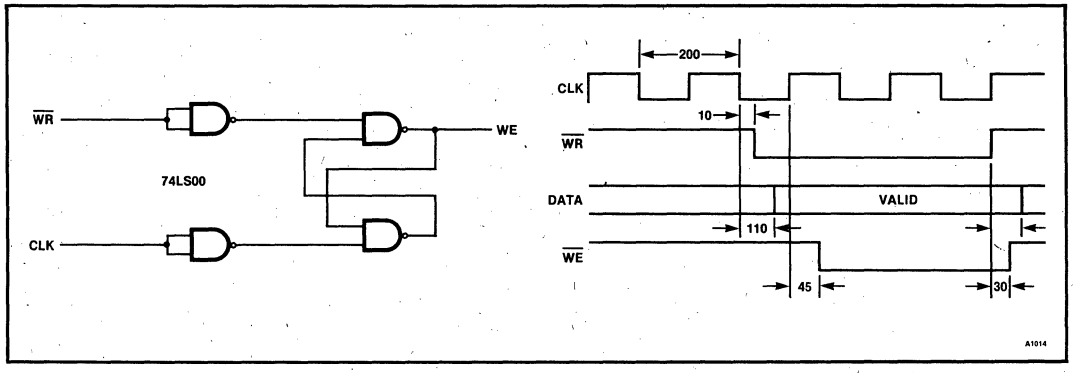

Figure 23. Simplified Write Enable Circuitry

provide a $\overline{CE}$ to the 2186. After completion of the external data memory fetch, P2.7 will revert to its preset value of "1".

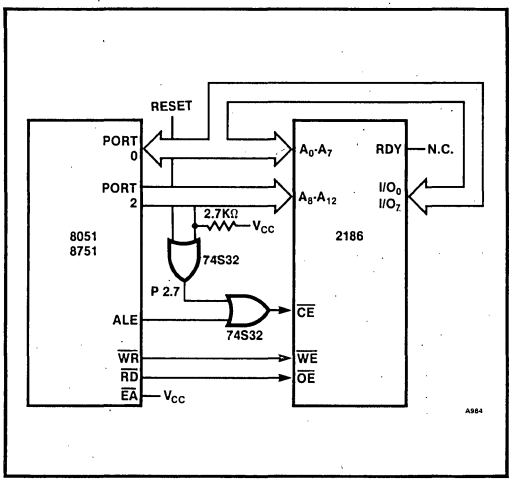

Figure 24. Asynchronous 8051 System

Note that a pull-up resistor is used to ensure that P2.7 will return to a "1" before the next trailing edge of ALE. Timings on the ALE are specified so that all $\overline{CE}$-related parameters on the 2186 are guaranteed, including address setup (TAVEL) and hold times (TELAX), and $\overline{CE}$ high time (TEHEL). The $\overline{RD}$ and $\overline{WR}$ outputs of the 8051 are tied directly to the $\overline{WE}$ and $\overline{OE}$ inputs to the iRAM. Data to be written is guaranteed to be valid before the leading edge of $\overline{WR}$ for the 8051. This provides the leading edge write needed by the 2186/87.

Although a RDY input does not exist for the 8051, a 2186 can still be used for data memory. At 8 MHz the 8051 does not require data back from the data memory until 800 ns after the trailing edge of ALE. The 2186-25 specifies worst case access time at 675 ns from the trailing edge of $\overline{CE}$, which in this system, corresponds to ALE. Even if the 2186 is just starting a refresh cycle when the 8051 requests an access, it will still have time to complete the refresh cycle, and access valid data by the time the 8051 requires it. Note that during RESET, $\overline{CE}$ is kept high to satisfy the power-up requirements of the 2186.

The access time required of program memory is somewhat faster than that needed for data memory. Because of this, the 2186 cannot be used in an asynchronous refresh mode as program memory for a full speed system. However, operation could be guaranteed if the system clock were slowed down.

The synchronous 2187 iRAM can be used as program storage for an 8051 running at 10 MHz by utilizing a method known as clock stretching. The circuitry, as shown in Figure 26, allows the 8051 clock to be stopped in a high state whenever the 2187 requires a refresh cycle. This stretched period is performed at the beginning of a cycle while ALE is high.

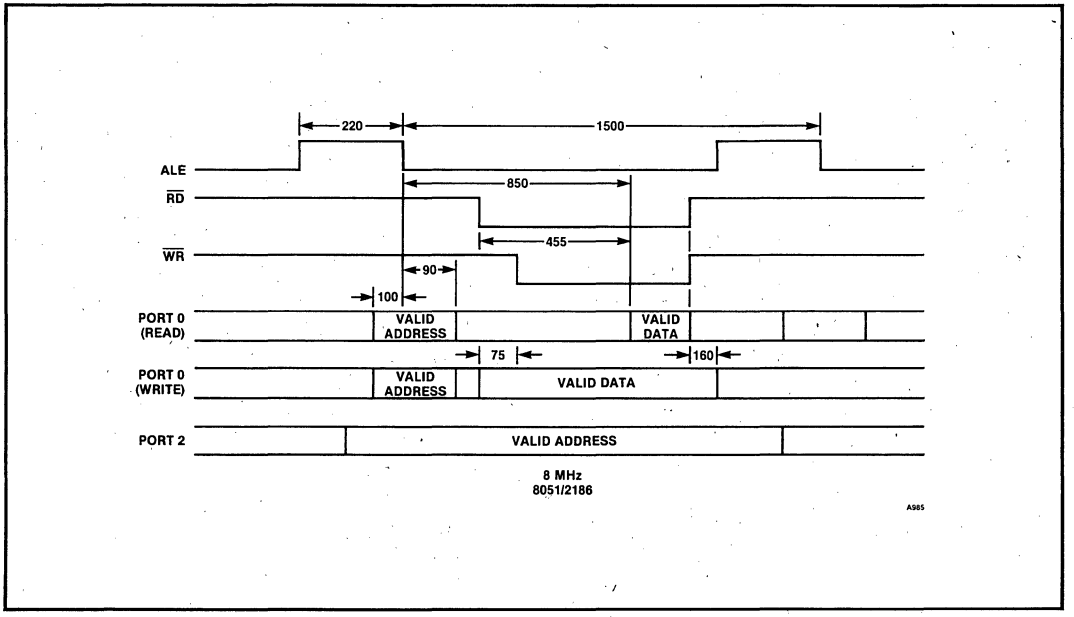

Figure 25. Asynchronous 8051 System Timing

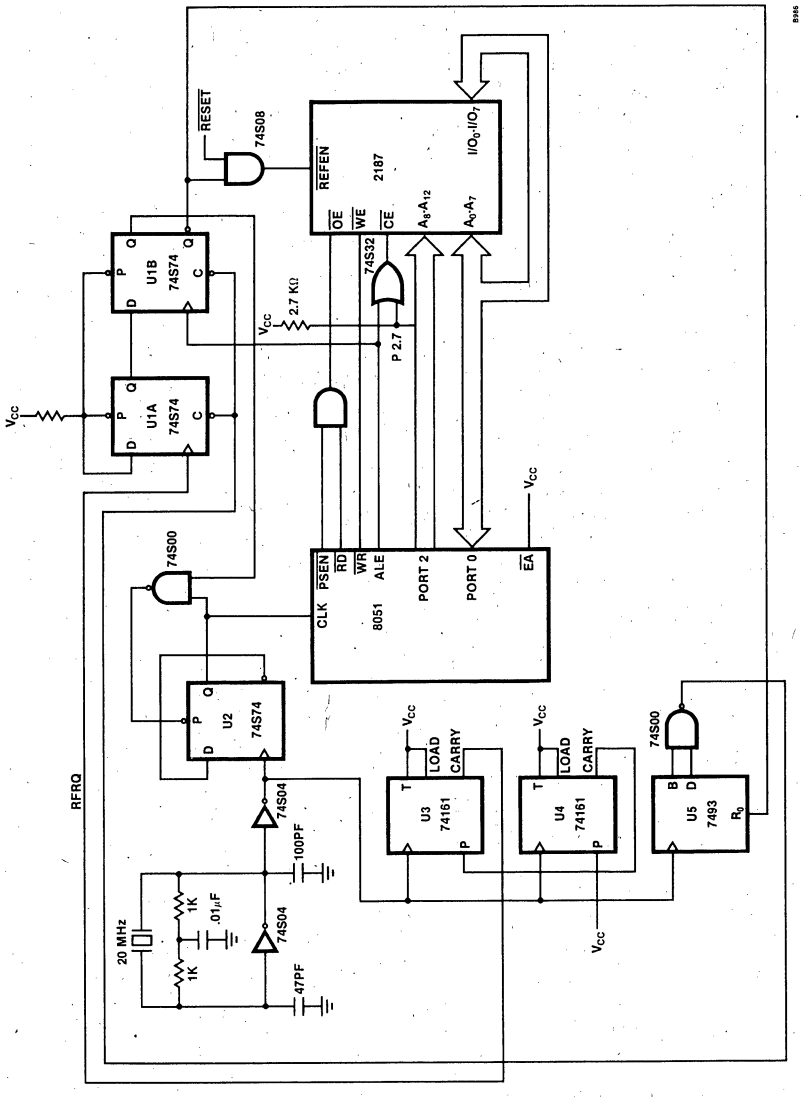

Figure 26. Synchronous 8051 System

Operation of the clock stretching circuitry is straightforward (Figure 27). Under normal operation, U2 acts as a frequency divider for the clock. U3 and U4 count clock pulses, and when a full count occurs, a refresh cycle request is issued (RFRQ). This request sets U1A. On the next high transition of ALE, this request is clocked into U1B, where it causes $\overline{\text{REFEN}}$ to become active. A refresh cycle within the 2187 begins at this time.

At the same time that $\overline{\text{REFEN}}$ becomes active, U5 is released from a clear state to start counting clocks, acting as an interval timer to allow time for the refresh cycle to occur.

On the first high transition of the system clock after U1B is set, U2 will be preset, maintaining the already high state of the clock. This high level is maintained until U5 has counted 10 clock cycles, at which point it acts to reset the clock stretching circuitry and allow the clock to return to a toggling condition.

The clock stretching circuitry used in this system could be utilized to a greater extent than just handling iRAM refresh cycles. For example, it might be useful for some type of DMA operation, or for use with slow peripherals. Also note that no address latches are needed with this system. To satisfy the power-up requirements of the 2187, $\overline{\text{REFEN}}$ must be held low for 100 μsec after V_{CC} is within its specified value. This is accomplished by driving $\overline{\text{REFEN}}$ low during RESET.

In a typical operation, a down-loader program would reside onboard the 8051 in PROM. This program would write program instructions into data memory. These instructions could then be "fetched" out of the same memory which would now be acting as program storage. This overlaying of program and data store is accomplished by allowing either $\overline{\text{PSEN}}$ or $\overline{\text{RD}}$ to enable the 2187 for a READ. Thus, it is possible to create a intermixed data and instruction field.

5.2 8088/2186 8-Bit Microprocessor Design Example

An example of an 8088/2186 iRAM design is shown in Figure 28. The 8088 is connected in a straightforward manner to the 2186 iRAM array. The low order addresses are latched from the multiplexed address/data bus of the CPU by ALE and are connected to the array. The CPU $\overline{\text{RD}}$ provides $\overline{\text{OE}}$ for the iRAMs while the $\overline{\text{MWTC}}$ from the 8288 bus controller serves as the $\overline{\text{WE}}$ for the memory. A stable chip select is generated by circuitry enclosed within the dashed lines. This circuit runs without WAIT states at 5 MHz using the 250 ns 2186-25.

5.3 8086/2186 16-Bit Microprocessor Design Example

The 5 MHz min mode system shown in Figures 29 and 30 depicts a typical interface of 2186 iRAMs with an 8086 16-bit microprocessor. With this arrangement, up to 128K words can be addressed.

To guarantee a stable $\overline{\text{CE}}$, the first interface circuit described in Section 4 is used. The output of this dual J-K flip-flop arrangement is used to enable the 8205 $\overline{\text{CE}}$ decoder.

A False Memory Cycle (FMC) is generated by this circuit during byte write cycles because both devices in the 16-bit word receive $\overline{\text{CE}}$, but only one device (or byte) receives a $\overline{\text{WE}}$. The other device enters an FMC without any consequences at the system level.

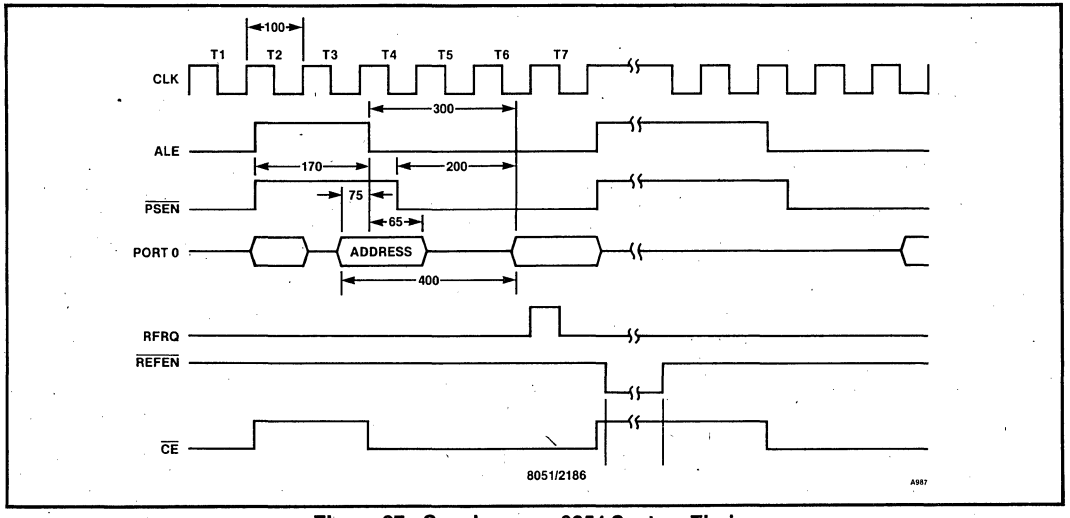

Figure 27. Synchronous 8051 System Timing

In min mode, the 8086 does not guarantee that valid data is present before the leading edge of $\overline{WR}$. A technique to delay this edge in order to provide the iRAMs with a properly timed $\overline{WE}$ must be included in the system. The cross coupled NAND arrangement described in Section 4 is used to provide both leading and trailing edge write compatibility.

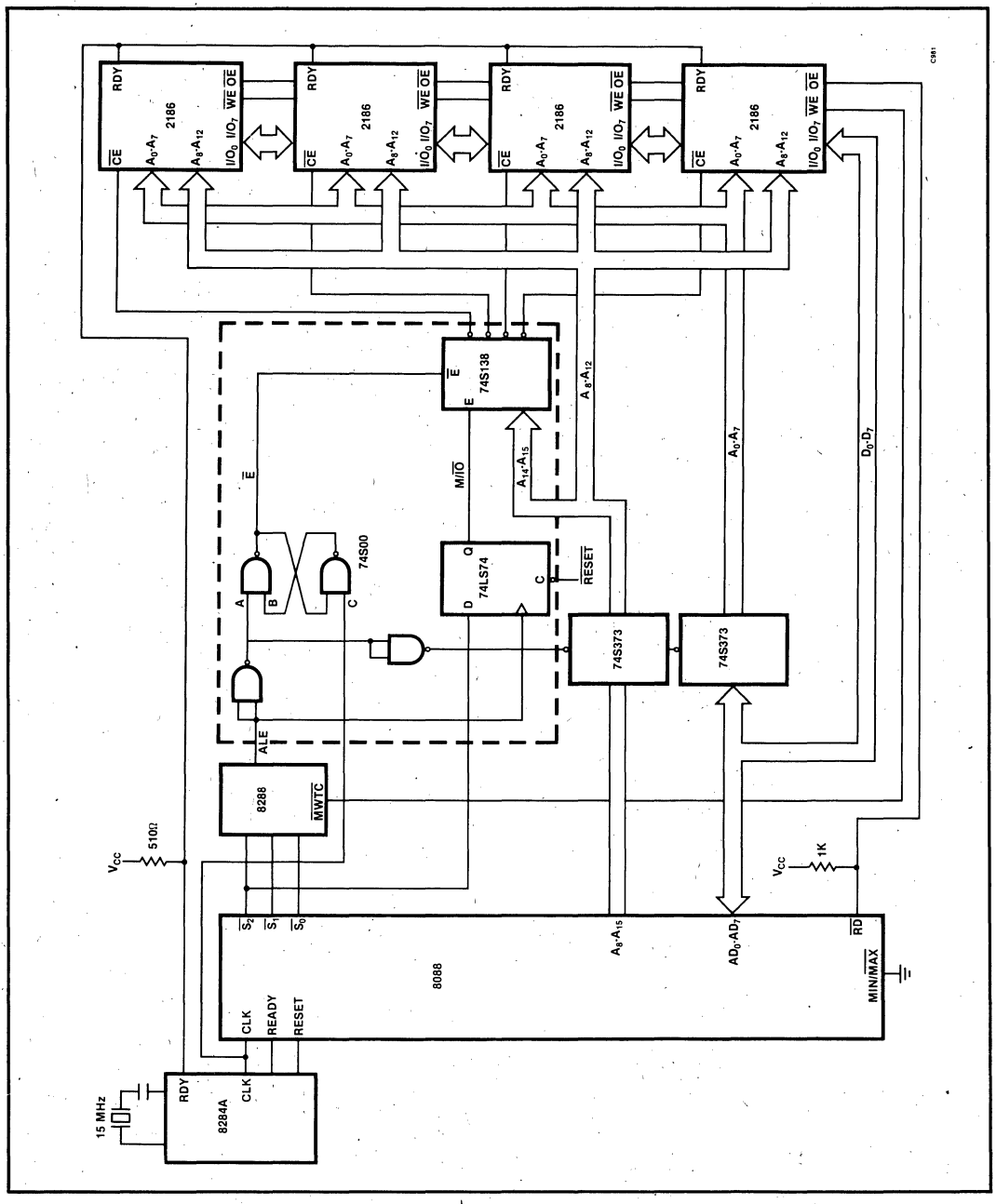

Figure 28. 8088/2186 Microprocessor System

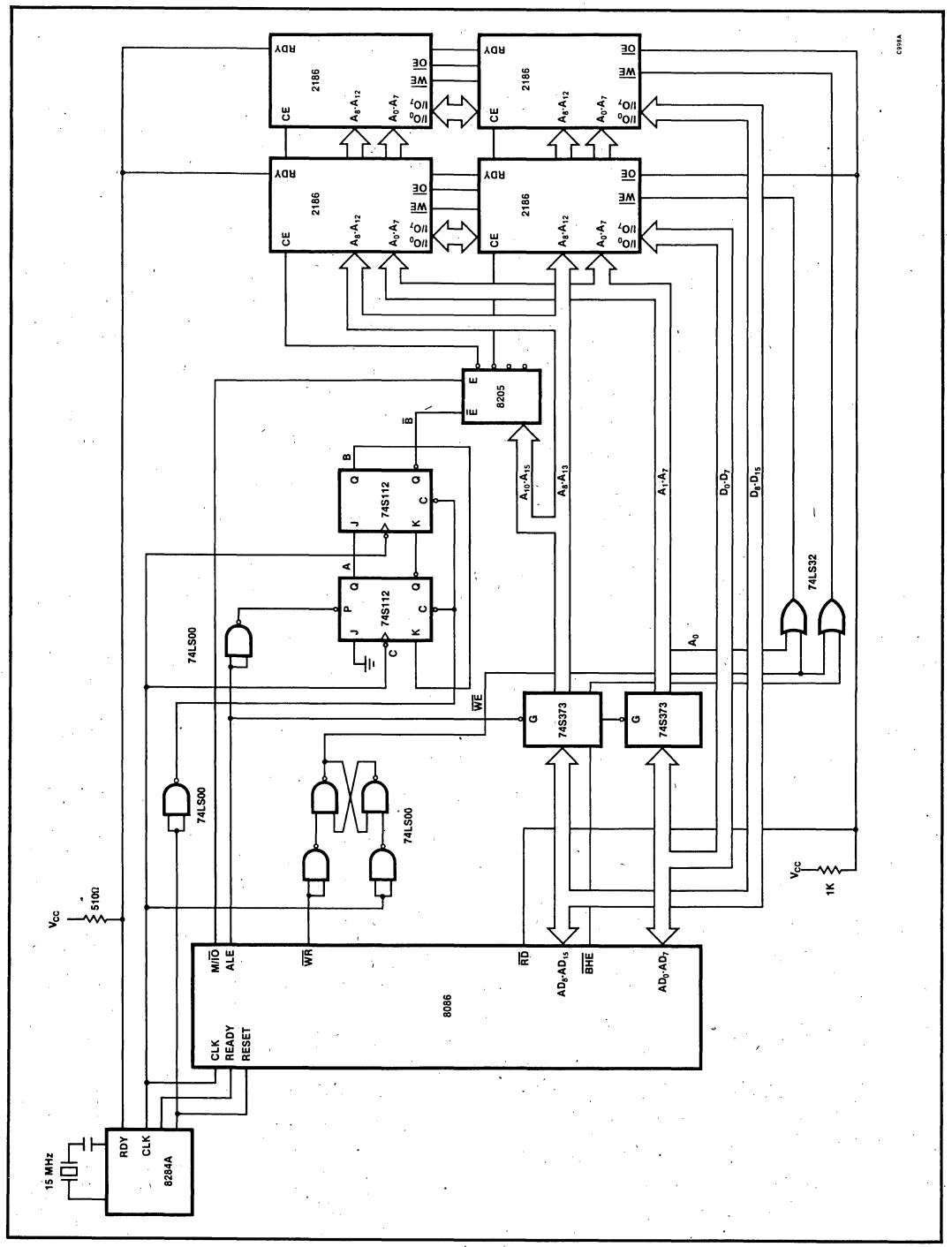

Figure 29. 8086 Min Mode System

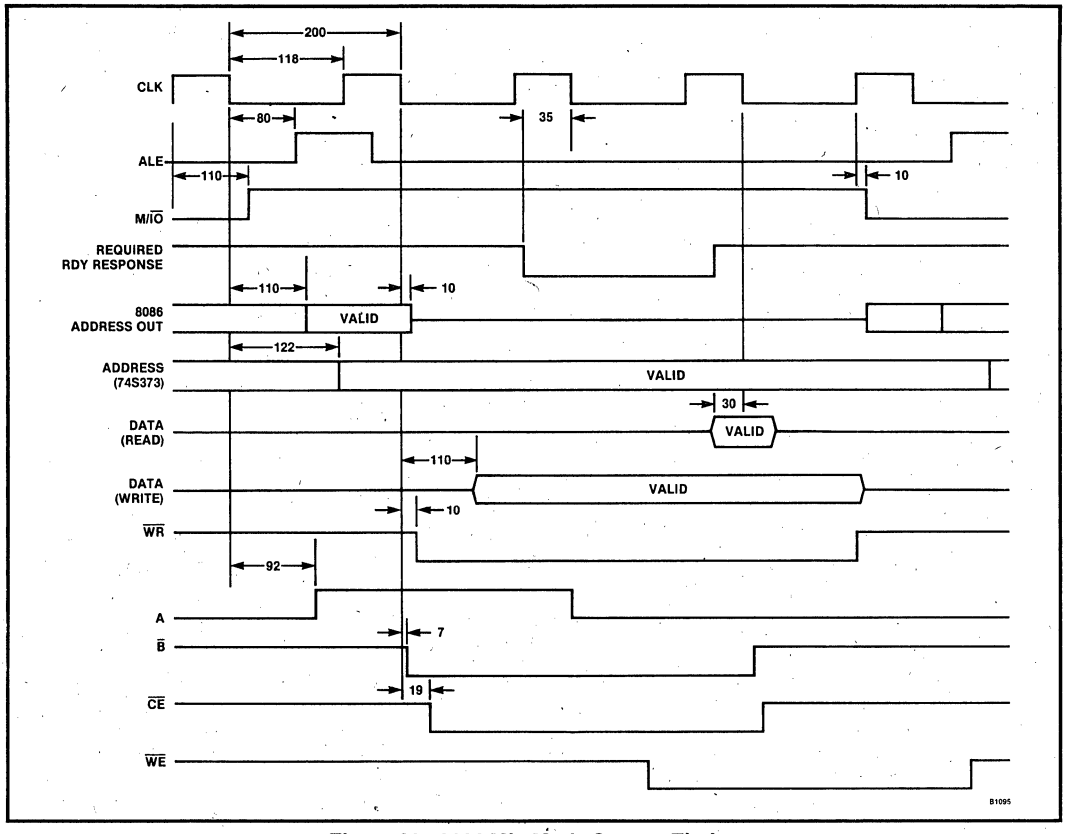

Figure 30. 8086 Min Mode System Timing

If an 8086 max mode system is to be used, the $\overline{WE}$ delay circuitry is not needed. In this case, the normal $\overline{WR}$ provided by the 8288 bus controller meets the leading edge write requirement. A diagram is shown in Figure 31. Note that a D-type flip-flop is used to latch $\overline{S2}$. This is important, because during certain 8086 operations, such as execution of a software HALT, $\overline{S2}$ is not guaranteed to remain valid up to the trailing edge of ALE. To overcome this, $\overline{S2}$ is latched on the leading edge of ALE, as done here.

5.4 Graphics Example

All of the applications examples presented thus far are non-specific; that is, all demonstrate how to connect the iRAMs to various microprocessors in the most general terms without regard to the total application. The design that follows shows the 2186/2187 iRAM in a specific application: a color graphics display memory.

In this example (Figure 32), the color display resolution is 65,536 (256 × 256 pixels) × 4 bits. The four bits select the color of the pixel by addressing a color lookup and video priority table. This programmable table permits up to 16 colors (out of 256 possible) per display frame. It also assigns priority. For example, a red disk crosses a green on the display. Does the red cross in front of the green disc, the green in front of the red, or does the area of the overlap become yellow? The priority encoding assigns answers to these questions.

By industry standards, this 256 × 256 pixel display has low-end to medium display resolution. For those unfamiliar with the capabilities at this level, visit a local video game parlor and examine some of the dazzling displays on the state-of-the-art video games such as Williams Electronics Defender. Advanced machines such as this are only beginning to approach this display density.

The iRAM used in this example is the synchronous 2187. Due to the sequential addressing scheme of video displays, video memory typically requires no additional circuitry for refresh. The 2187 is no exception, and in this design the $\overline{REFEN}$ pin is tied high. The sequential scanning by the video address generator automatically refreshes the internal array of the iRAM.

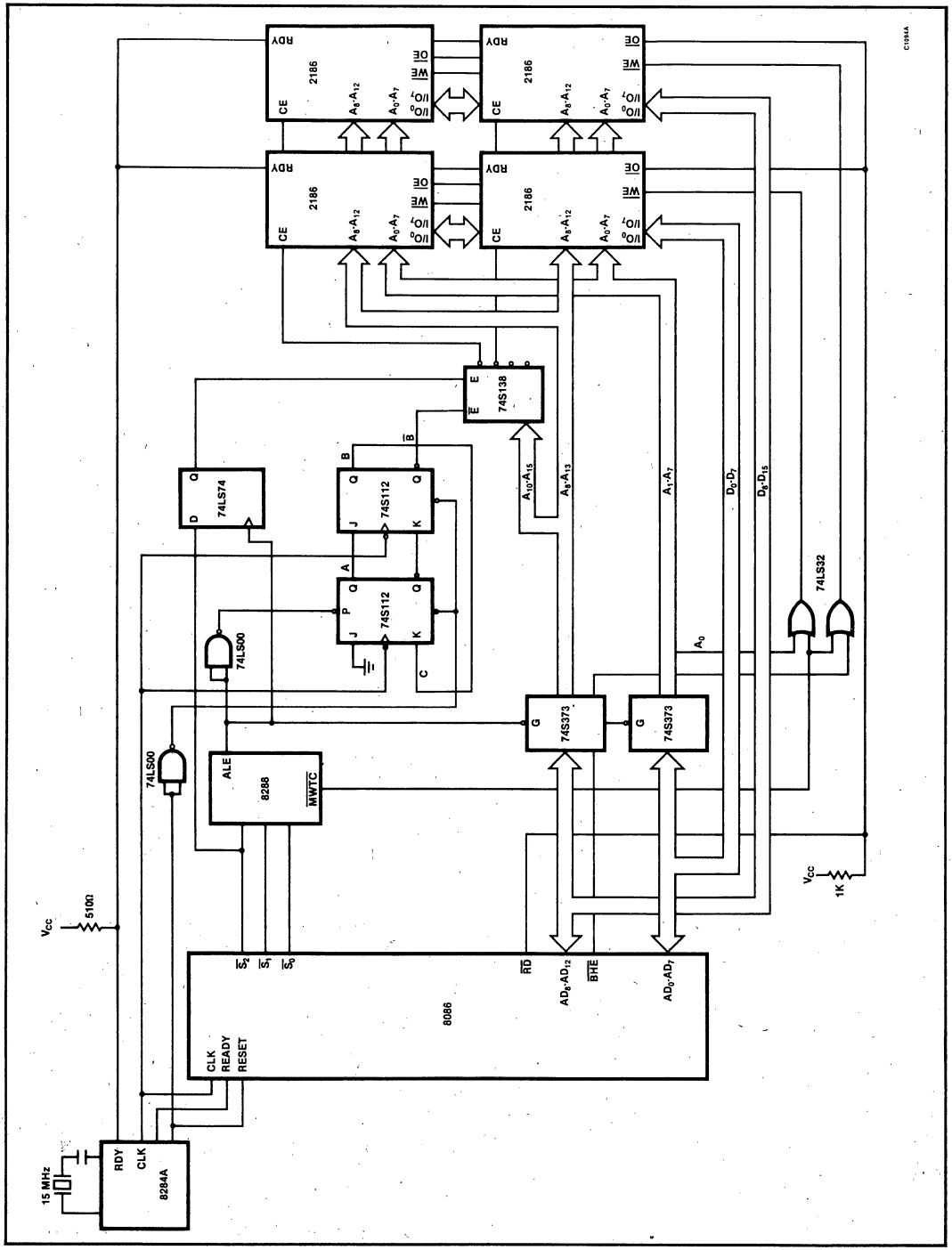

Figure 31. 8086 Max Mode System

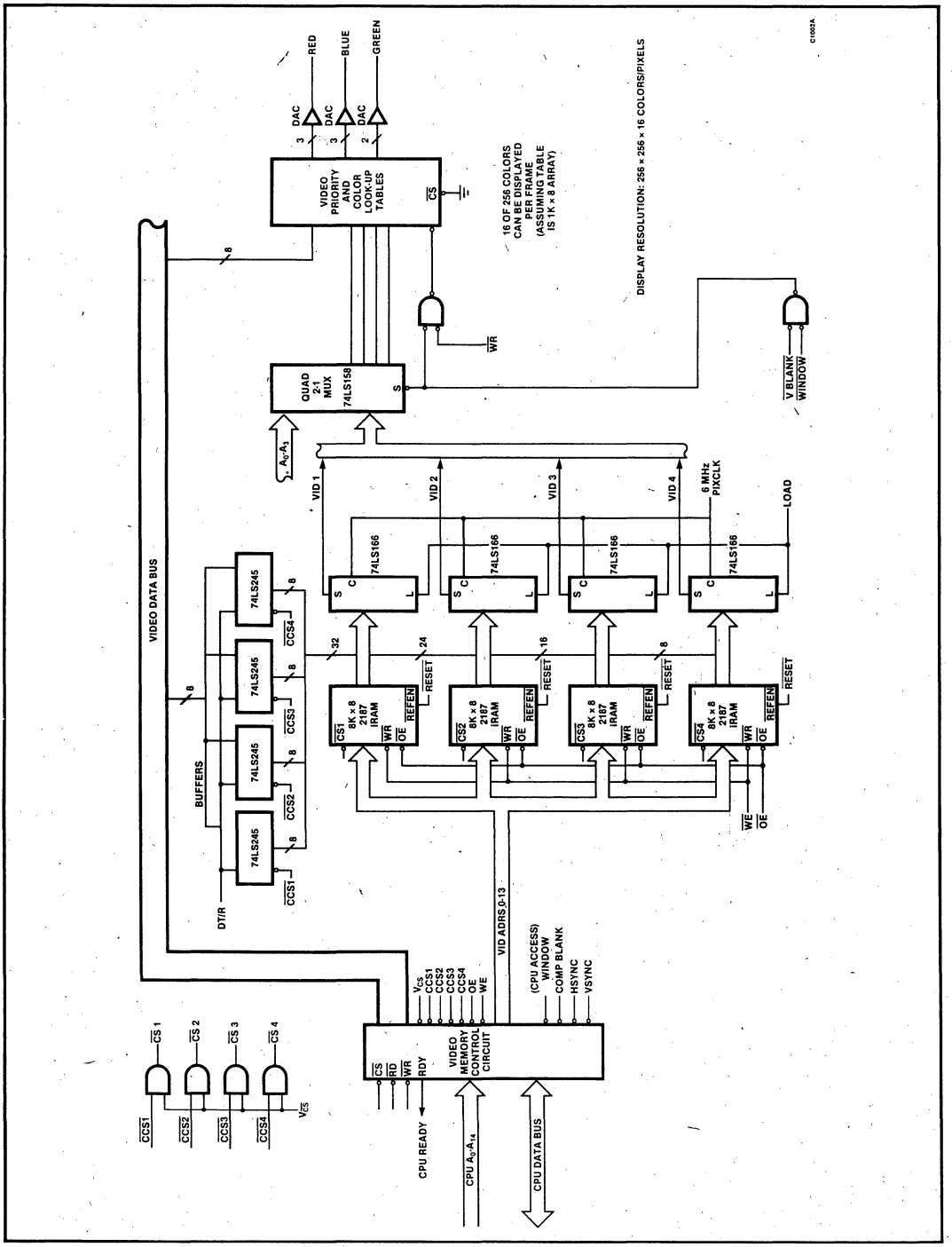

Figure 32. Graphics System

Figure 32 is a simplified diagram. A detailed analysis of the circuit and timing will not be discussed. Briefly, the circuit functions as follows:

CPU addresses A_{14} and A_{15} are decoded to generate one of four iRAM chip selects so that the (assumed 8-bit) CPU can read or write information to the individual memory planes (iRAMs). These chip selects are gated so that all four iRAMs can be simultaneously enabled by the V_{CS} signal from the video timing circuitry. A similar circuit (not shown) would allow $\overline{OE}$ for the iRAMs to be generated by either the CPU or the video timing generator. The iRAM addresses are generated by multiplexing the CPU addresses with video timing addresses. The 32-bit output from the iRAMs is loaded into four 8-bit shift registers and are serially shifted out as four bits of video information used to address the color lookup table. The four lines (Vid_1-Vid_4) are multiplexed with CPU addresses A_0-A_3 to create the actual addresses of the lookup table. Comprised of two 2148H RAMs, the eight data lines of the lookup table are directed to three digital-to-analog converters for generating 16 of 256 different display colors.

Due to the byte-wide organization of the iRAMs, there is plenty of time between video read cycles to allow CPU access to the memory. With a pixel rate of 6 MHz, the byte-wide iRAM has a video read rate of 6/8 MHz or once every 1.33 microseconds. Only 350 ns of this time is needed for a video read cycle. The balance of the time (approximately 1 μsec) can be used by the CPU to access the memory. This interleaving of CPU cycles with video timing cycles, combined with allowing the CPU unrestricted access to the memory during both horizontal and vertical blanking (retrace) periods permits the real time screen update required in an animated display.

5.5 External Refresh Systems

5.5.1 BURST REFRESH

Figure 33 shows an example of a burst mode refresh controller. Timings for this system are shown in Figure 34. To ensure data integrity for a 2187, $\overline{REFEN}$ must be strobed at least 128 times in each 2 ms period. After each high-to-low transition of $\overline{REFEN}$, one cycle time must be allowed before $\overline{REFEN}$ (or $\overline{CE}$) again becomes active.

The system shown in Figure 33 accomplishes refresh by interrupting the processor once each 1.63 ms (200 ns clock period divided by 8192). Upon acknowledgment of this interrupt, $\overline{TEST}$ is driven high, allowing $\overline{REFEN}$ to be generated once every three clock cycles. $\overline{TEST}$ is also routed back to the $\overline{TEST}$ pin of the 8086 to indicate that a burst is in progress. The 8086 samples the state of the $\overline{TEST}$ pin and loops in an idle state until the $\overline{TEST}$ goes low. This is accomplished using the WAIT instruction.

Two 4-bit counters are used to count the $\overline{REFEN}$ pulses. After 128 pulses, the count goes high. On the next rising edge of the system clock, $\overline{TEST}$ is reset to a zero, blocking any further $\overline{REFEN}$ pulses, clearing the counters, and signaling to the processor that the burst is complete. Note that one non-access cycle should be inserted after $\overline{TEST}$ is set low to ensure that sufficient time has been allowed for the last refresh cycle to complete.

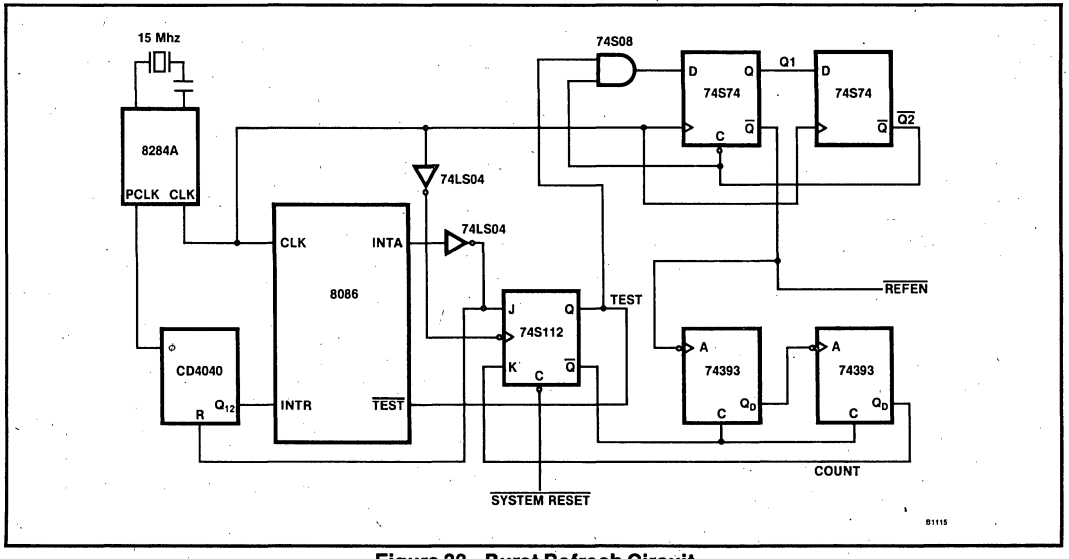

Figure 33. Burst Refresh Circuit

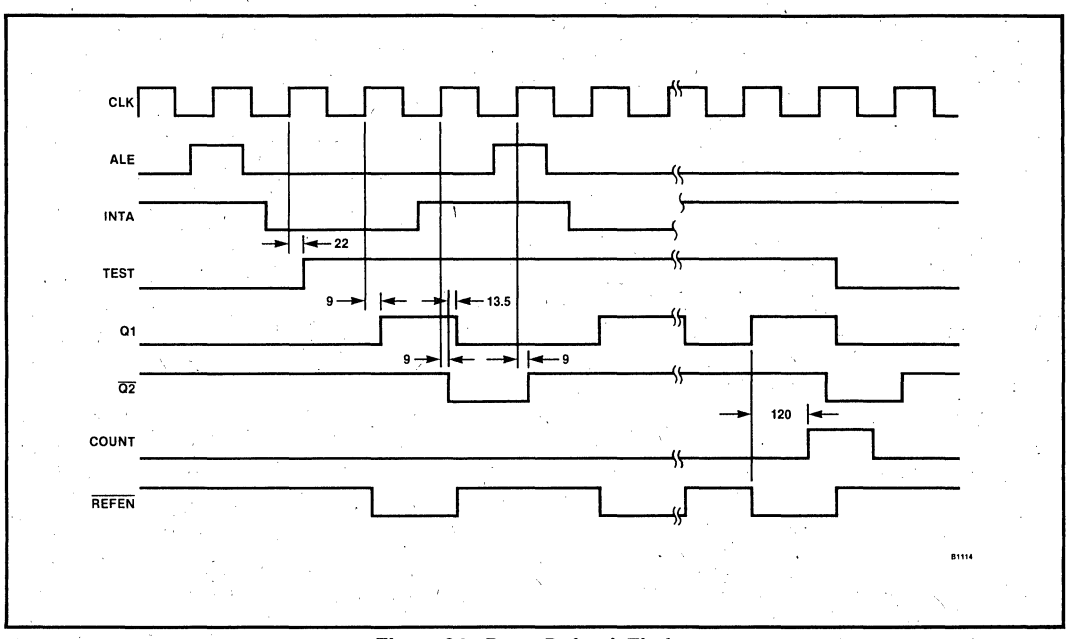

Figure 34. Burst Refresh Timing

5.5.2 SYNC REFRESH SYSTEM

The system in Figure 35 represents one way in which synchronous refresh could be employed using the 2187. In this configuration, memory is divided into four banks, selected via the two least significant addresses. To ensure data integrity, each of the four banks must receive 128 $\overline{\text{REFEN}}$ pulses every 2 ms. In this system if any one bank is accessed, each of the other three banks receives a refresh pulse. Minimum cycle time cannot quite be attained because the cycle time for the refresh cycle is the same as that for an access cycle, and the fact that a one-gate delay exists between $\overline{\text{CE}}$ to one device and the $\overline{\text{REFEN}}$ to the others. At least 16 ns must be added to the minimum cycle time of 425 ns. This number is derived by taking the propagation delay difference between a "fast" 74155 and a "slow" 74155, and adding the maximum delay through a 74S11. This gives the $\overline{\text{CE}}$ to $\overline{\text{REFEN}}$ delay time. This extra delay is not really critical in most systems; the minimum cycle time for a 5 MHz 8086 is 800 ns.

With the circuitry described, data integrity would be jeopardized if one bank were accessed consecutively too many times, since the accessed bank would receive no $\overline{\text{REFEN}}$ pulses. Assuming a 500 ns cycle time, one bank would have to be accessed at least 30 consecutive times to jeopardize data. This is the worst case. In actual operation, consecutive accesses to one bank could be many more than this, as long as operation during any 2 ms period provides 128 $\overline{\text{REFEN}}$ pulses to all banks. Due to the nature of bank selection used (A0:A1 decoding), more than a couple of consecutive accesses to any one bank are highly unlikely.

One caution to note, however, has to do with power-down refresh. If $\overline{\text{REFEN}}$ is kept low for longer than one timer period, the timer will begin to time out. In this event, a period of time (RFHEL) must be allowed before $\overline{\text{CE}}$ can go low again after $\overline{\text{REFEN}}$ returns high. This is to ensure, that if a timer initiated refresh cycle started just as $\overline{\text{REFEN}}$ returned to a high state, it will have time to complete before an access cycle is started.

6 SYSTEM CONCEPTS

6.1 System Reliability

New applications for microprocessor systems appear almost every day. They appear in microwave ovens, automobiles, word processors, home computers, video games, vending machines, lighting controls, medical equipment, etc. The list goes on and on. Failures on these systems cover equally broad ranges: acute annoyance (such as losing your last quarter to the coffee machine), financial loss (a double debit is added to your bank statement by an electric teller machine), and life threatening system failures (the electronic carburator control on your car fails, opening the throttle wide open).

In many applications, reliability is important enough to be designed into the system. The computer memory system is one of the system components for which reliability is important. Also it is one of the few system elements which can be easily designed to enhance its reliability. Since memory system reliability is inversely proportional to the number of devices in the system, a system of a given size should be designed with as few components as possible. For example, a 32K byte system could be designed with sixteen 16K 2118 DRAMs. The system MTBF (Mean Time Between Failures — the "up" time of a system) could be calculated from the combined device soft and hard error rates (See Intel Application Note AP-73 *"ECC #2 Memory System Reliability With ECC"* for a model to calculate system MTBF's). The point is that, whatever the calculated system MTBF, the 2186 will be several times more reliable in a system due to the lower device count.

A few example calculations are tabulated in Table 1. Essentially what is shown is what the maximum acceptable device soft error rate is for a specified system MTBF. For example, if a design using 8K × 8 RAMs requires a memory system MTBF for two years, and the system size is 16K bytes, then the design allows a device with a soft error rate of 3.1%/1K-hrs. The 2186/87 soft error rate goal is more than an order of magnitude better than that! From the chart it can be seen that a 64K byte extra-reliable memory system with a 10 year MTBF requires a device with a soft error rate or 0.15%/1K-hrs. Clearly the 2186/87 family of iRAMs is reliable over the entire spectrum of typical application memory sizes.

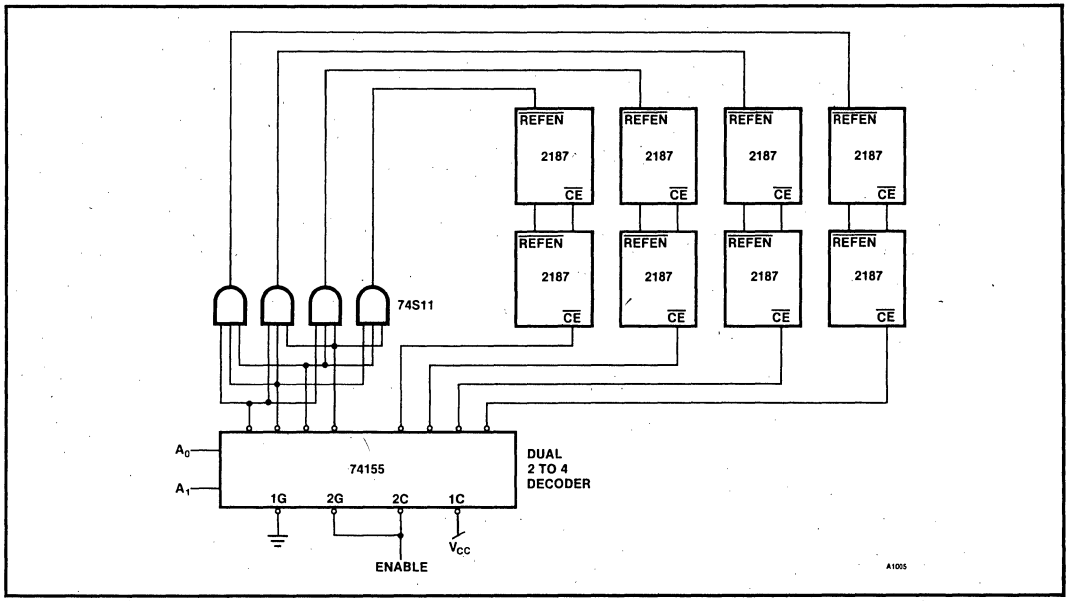

Figure 35. Synchronous Refresh Scheme

Table 1. 2186/87 SER Data

No. of Syst. Rows	No. of Dev.	Sys. Size	Eff. Cycle Time*	Maximum Allowable SER (%/K-Hrs.)			
				1 Yr. MTBF (8800 Hrs.)	2 Yrs. MTBF (17600 Hrs.)	5 Yrs. MTBF (44000 Hrs.)	10 Yrs. MTBF (88000 Hrs.)
1	1	8K	7.00	11.34	5.66	2.25	1.12
2	2	16K	9.66	6.23	3.10	1.22	.60
3	3	24K	11.06	4.29	2.13	.84	.41
4	4	32K	11.93	3.27	1.62	.64	.31
8	8	64K	11.52	1.67	.82	.31	.15

* All times in microseconds
System has a 7μsec device cycle time.
Hard error rate = 0.02%/1K-Hrs.

6.2 Circuit Design Considerations

Integrating components into systems requires a keen awareness of basic concepts on the part of the designer.

Techniques for designing optimal performance memory systems have been thoroughly covered in other literature. Two useful documents that cover these procedures are AP-74 *"High Speed Memory System Design Using the 2147H"* and AP-133 entitled *"Designing Memory Systems for Microprocessors Using the Intel 2164A and 2118 Dynamic RAMs."* There are essentially three areas of major concern in a memory system design:

- Timing delay calculations in the critical path (worst case timing analysis)
- Memory circuit trace layout
- Power distribution and decoupling

The following sections summarize these techniques as they apply to the 2186 and 2187 iRAMs.

6.2.1 DELAY CALCULATIONS

All memory designs require a timing analysis to ensure proper operation and compatibility of the memory and the processor. Timing skews, capacitive delays and propagation delays all have to be accounted for in a proper analysis. Propagation delay design rules for TTL are furnished in the manufacturer's data book. The maximum delay is the data book maximum and the typical delay (usually useless for design) is the data book typical. Intel has determined in work with TTL device manufacturers that the minimum propagation delay is ½ the data book typical value.

Skew is defined as simply the difference between the maximum and minimum propagation delays through devices in a parallel path. Figure 36 is a simple example. Best case propagation of signal A is 6 nanoseconds versus worst case delay of signal B which is 16 nanoseconds. This condition equates to 10 nanoseconds of skew (Figure 37) which adds directly to system access or cycle time. The worst case number of 16 ns would be used for timing analysis in this type of delay calculation; however, often the best case is the most important. For example, as in Figure 38, the skew of concern deals with the best case arrival of a write pulse versus worst case arrival of data to a memory device.

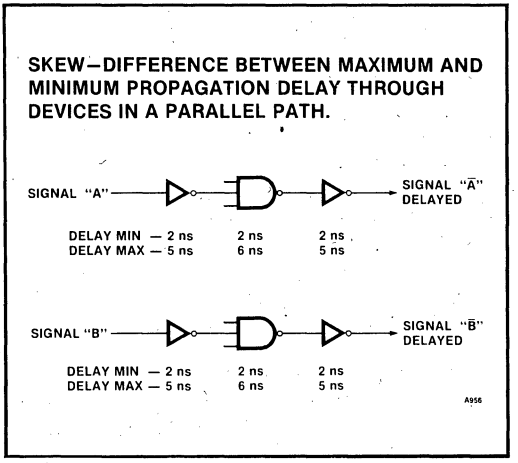

Figure 36. Skew

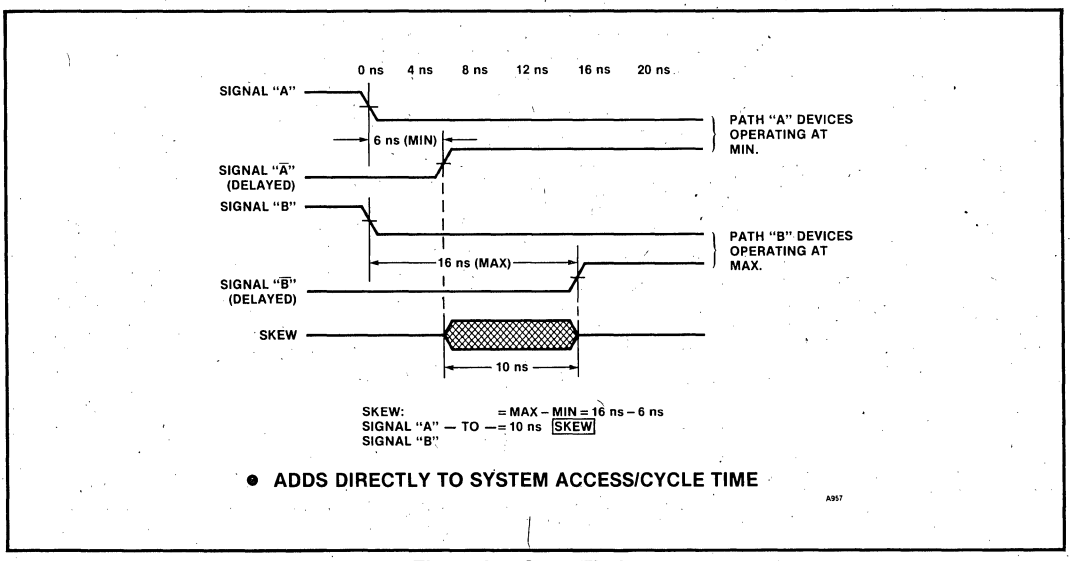

Figure 37. Skew Timing

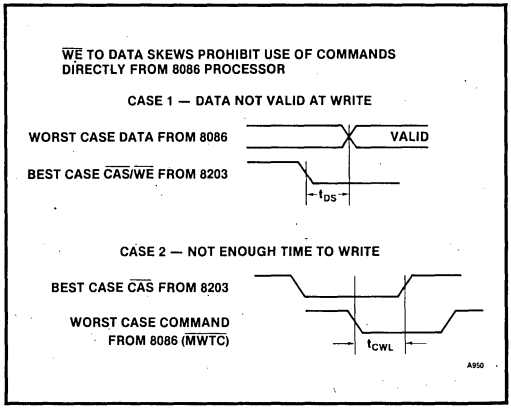

Figure 38. Worst Case Timing

Unbalanced capacitive loading on address or control line drivers also contribute to skew. Capacitance contributes to risetime degradation on these signals. The unbalanced loading causes differing rise times as shown in Figure 39. The different rise times reach a logic threshhold at different times, contributing to skew. In all of these examples, skew contributes to the overall delay, and the goal of the designer is to minimize these skews. A few simple rules will help to achieve this in 2186/87 memory system design:

- Select logic gates for minimum delay per function
- Place parallel paths in the same package (device to device skew is much less within same package - 0.5 ns max for STTL)
- Balance the output loading of device drivers to equalize capacitive delays.

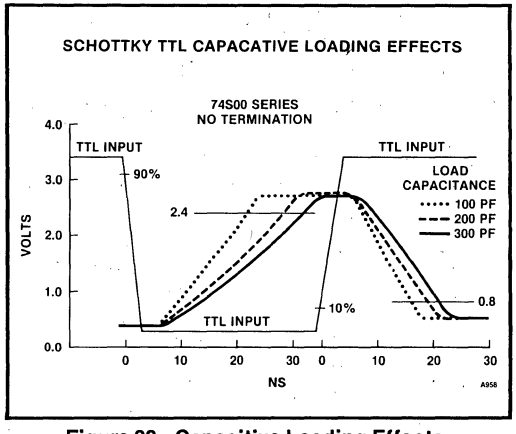

Figure 39. Capacitive Loading Effects

As previously stated, capacitance contributes to signal risetime degradation. To determine the delay due to capacitance, use the following standard derating factors:

Schottky TTL = 0.05 ns/pF
Low Power Schottky TTL = 0.1 ns/pF
Standard TTL = 0.75 ns/pF

Add up all of the capacitance connected to a driver, including the circuit-printed trace capacitance of 2 pF per inch, subtract out the manufacturer's capacitance drive specification, (typically 15 pF) then multiply this capacitance by the derating factor for the driver. This net result is the additional delay due to capacitance. The equation is:

$$D_C = [\Sigma C_{IO} + \Sigma C_{PCB} - C_{SPEC}]T_D$$

where: D_C = delay due to capacitance
ΣC_{IO} = sum of all input/output connections attached to driver
ΣC_{PCB} = 2 pF × number of inches of circuit trace attached to driver
C_{SPEC} = specified drive capacitance of driver
T_D = capacitive derating factor

6.2.2 TRACE LAYOUT

Address lines need to be kept as short and direct as possible. Route address lines in a comb-like fashion from a central location. Routing control and address signals together from a centralized board area will also minimize skew.

Allow for proper termination of all address and control lines because these circuit traces are actually transmission lines. A series resistor close to the driver is the recommended termination technique. Thirty-three ohms is a good typical value, although actual values are usually determined empirically. Figure 40 shows P.C.B. artwork that embodies these rules as well as proper power and ground gridding with decoupling as described in the following section.

6.2.3 POWER SUPPLY DISTRIBUTION AND DECOUPLING

Ground and power busses can contribute to excess noise and voltage drops if not properly structured. The power and ground network do not appear as a pure low resistance element but rather as a transmission line, because the current transients created by the RAMs are high frequency in nature.

Transient effects can be minimized by adding extra circuit board traces in parallel to reduce interconnection inductance. Extrapolation of this concept to its limits will result in an infinite number of parallel traces, or an extremely wide low impedance trace, called a plane. Arranging power and ground voltages by plane provides the best distribution; however, correct gridding can cost effectively approximate the benefits of planar distribu-

tion by surrounding each device with a ring of power and ground traces (Figure 40).

Consider two aspects of the memory device that contribute to power system noise: the active/standby power modes of the RAMs, and the drive requirements of the data I/O buffers. In a typical microprocessor-based system, address space is divided into blocks of RAM, ROM/EPROM, and I/O. When the microprocessor is not accessing a given RAM, the RAM is usually deselected and in a power standby mode. When a previously unselected RAM is selected, a large current surge is experienced. Because the connections supplying power to the device will involve resistance and inductance, a voltage variation will occur in association with the current surge in accordance with the equation:

$$V = Ri + Ldi/dt,$$

where V = instantaneous voltage,
L = inductance,
R = resistance,
and i = instantaneous current

Because a RAM may be selected and deselected hundreds of thousands of times a second, the transient noise generation is significant and must be dealt with during design.

Another factor that contributes to current surges are the drive requirements of the memory devices data I/O buffers. Consider first an I/O buffer outputting a logic one. To accomplish this, the buffer must supply a current to charge the capacitance of the line that it's driving to a logic one level. This operation places a higher current requirement than normal on the V_{CC} bus. Conversely, if the I/O buffer is outputting a logic zero, it must discharge all of the capacitance on the line to ground. This produces a current surge to the ground bus, possibly raising the local V_{SS} potential above ground during the transient.

The solution to this problem is to use a solid plane V_{CC} and ground bus on a P.C. board or use a proper power and ground grid combined with adequate decoupling.

Adequate decoupling is also important in circuit design to minimize transient effects on the power supply system. For best results with the 2186/87, decoupling capacitors are placed on the memory array board at every device location (Figure 40). High frequency 0.1 μF

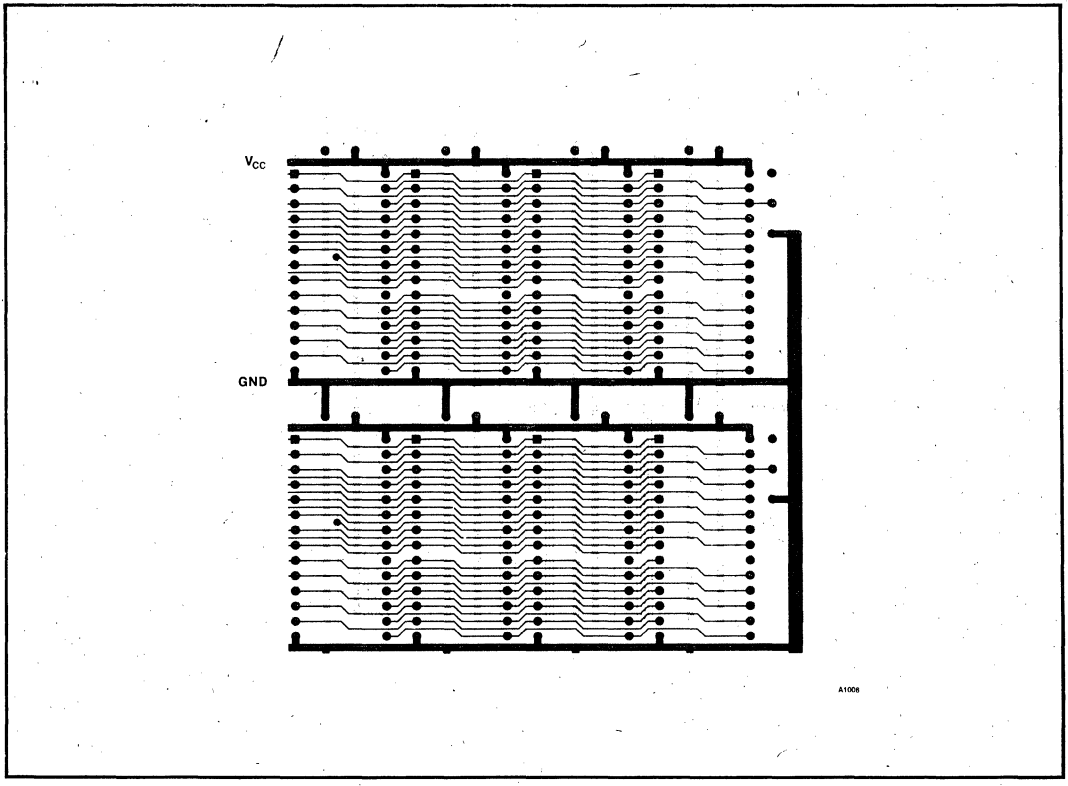

Figure 40. Example of Power and Ground Gridding

ceramic capacitors are the recommended type. Also included should be a large bulk decoupling capacitor in the 50 to 100 μF range, placed where power is supplied to the memory system grid. In this arrangement, each memory is effectively decoupled and the noise is minimized because of the low impedance across the circuit board traces.

7 SUMMARY

Intel's iRAMs provide a new approach to memory design that allows the system designer to take advantage of DRAM density, power consumption, and price without the added cost of designing the refresh control circuitry. The 2186 and 2187 are the premier members of this new byte-wide product family, designed for flexible operation in virtually any microprocessor memory system. By comforming to Intel's universal memory site concept, these iRAMs are compatible with a wide variety of byte-wide memory devices including SRAMs, EPROMs, and E^2PROMs.

In summary, Intel provides another innovative memory product, the 2186/87 iRAMs — basic building blocks for microprocessor memory solutions.